Test and Evaluation of CF-18 G-Valve Prototype

Final Report

Ziren Lu, Matt Morel, Alex Kapps, Aaron Galluzzi

Engineering Services Inc.
1 Bedford Rd, Unit 5
Toronto, Ontario Canada M5R 2J7

Contract Number: W7711-007631

Defence and Civil Institute of Environmental Medicine

Final Report
DCIEM-CR 2002-006
October 29, 2001

Test and Evaluation of CF-18 G-Valve Prototype

Final Report

Ziren Lu, Matt Morel, Alex Kapps, Aaron Galluzzi

Engineering Services Inc
1 Bedford Rd, Unit 5
Toronto, Ontario Canada M5R 2J7

Project Manager Dr. Alex Kapps, tel: (416) 595-5519, ext. 232

Contract Number: W7711-0-7631

Contract Scientific Authority: William Fraser, Head, SAILSS Group, Defence and Civil Institute of Environmental Medicine, 1133 Sheppard Avenue West, PO Box 2000, Toronto, ON M3M 3B9

The scientific or technical validity of this Contract Report is entirely the responsibility of the contractor and the contents do not necessarily have the approval or endorsement of Defence R&D Canada.

Defence and Civil Institute of Environmental Medicine (DCIEM)

Final Report
DCIEM-CR-2002-006
October 29, 2001

Author

Ziren Lu, Matt Morel, Alex Kapps, Aaron Galluzzi
Engineering Services Inc.

Approved by

William Fraser
Head, SAILSS Group, DCIEM

Approved for release by

F.J. (Fred) Maggio
Head, Aerospace Life Support Sector, DCIEM

Administrative Information

Project Manager:	Dr. Alex Kapps, tel. (416) 595-5519, ext. 232 Engineering Services Inc 1 Bedford Rd, Unit 5 Toronto, Ontario Canada M5R 2J7
Contract Number:	W7711-0-7631
Contract Scientific Authority (CSA).	William Fraser, Head, SAILSS Group, DCIEM

© Her Majesty the Queen as represented by the Minister of National Defence, 2001

© Sa majesté la reine, représentée par le ministre de la Défense nationale, 2001

Abstract

This final report summarized the preliminary experimental testing program and statistical validation of novel countermeasures aimed at alleviating Push-Pull effects. The new countermeasures for active attenuation of Push-Pull effect involve applying mid-level Positive Pressure Breathing (PPB) during the Push phase in accordance with the specially formulated non-linear pressure schedules. To formally investigate the proposed countermeasures, a custom-designed statistical modelling framework ESIstat has been integrated using R language.

It has been statistically demonstrated in Tilt-table experiments that the application of medium-intensity (30mmHg) PPB during -Gz followed with the application of G-suit pressure and regular-intensity PPB during the transition to +Gz attenuates the Push-Pull effect and may prevent the subsequent decrease in a pilot's G-tolerance due to -Gz exposure. If the new advanced tests planned on the upcoming Push-Pull centrifuge at DCIEM substantiate the proposed protection strategy, it might potentially constitute an efficient and practical Anti-G countermeasure. The new protection can be readily incorporated into the existing Anti-G ensemble by simply retrofitting the PPB mask/jerkin regulator.

Résumé

Le rapport final résume le programme d'essai expérimental préliminaire et la validation statistique de contre-mesures innovantes visant à atténuer l'effet « pousser-tirer ». Les nouvelles contre-mesures comprennent l'utilisation de la respiration à pression positive (PPB) pendant la phase « pousser », conformément à des tableaux de pressions non linéaires spécialement formulés. Afin d'étudier de façon formelle les contre-mesures proposées, un modèle statistique clientélisé, ESIstat, a été intégré au moyen du langage R.

On a démontré statistiquement dans des expériences sur table basculante que l'utilisation d'une respiration à pression positive de moyenne intensité (30 mm Hg) au cours d'une accélération négative -Gz, suivie du gonflage normal de la combinaison anti-G et d'une respiration à pression positive d'intensité normale pendant la transition vers une accélération positive +Gz atténuait l'effet « pousser-tirer » et pouvait prévenir la diminution subséquente de la tolérance du pilote aux accélérations à cause d'un exposition aux accélérations négatives -Gz. Si les nouveaux essais évolués qui sont prévus dans la centrifugeuse « pousser-tirer » de l'IMCME corroborent la stratégie de protection proposée, cette dernière pourrait constituer une contre-mesure anti-G pratique et efficace. La nouvelle protection peut être facilement incorporée dans l'ensemble anti-G actuel simplement par l'ajout d'un régulateur de gilet et d'un masque PPB.

Executive summary

Current aircrew life-support systems provide a pre-programmed sub-optimal level of high-acceleration protection. The physiological responses of a pilot to the acceleration stresses seen in tactical aircraft are typically nonlinear and vary depending on the G-time history of the flight, especially the time at which the aircraft has spent in a relative negative Gz environment (i.e., Push-Pull effect). Advances in technology have allowed for the development of computer-controlled Anti-G valves that adjust/control G-suit inflation in real-time of flight based on Gz-history of an aircraft. The concepts of incorporating experts' experience and knowledge as well as real-time biofeedback into software for automatic real-time Anti-G protection control has been investigated at the previous stages of ESI's work. In 2000, DCIEM and ESI have performed an experimental study (Protocol L-239) to test some of these candidate nonlinear protection schedules during low-intensity Push-Pull transitions.

In this project, ESI has integrated an advanced statistical modelling framework ESIstat to investigate experimental data collected in Protocol L-239 study. This new statistical framework has been applied to analyze and statistically substantiate a new CF-18 G-valve control algorithm for active attenuation of Push-Pull effect by applying mid-level Positive Pressure Breathing (PPB) during the Push phase. To ensure compatibility with CF's requirements on formally reporting manned test results, the statistical framework implemented by ESI has been subsequently equipped with the automatic report generation capabilities. Furthermore, as a part of this project, the previously developed real-time electronic control algorithms for CF18 G-valve has been partially modified to ensure compatibility with the hardware specifications and test protocols used for manned trials on the US Air Force Laboratory Centrifuge.

It has been statistically demonstrated in Tilt-table experiments that the application of medium-intensity (30mmHg) PPB during -Gz followed with the application of G-suit pressure and regular-intensity PPB during the transition to +Gz attenuates the Push-Pull effect and may prevent the subsequent decrease in a pilot's Gz-tolerance due to -Gz exposure.

Sommaire

L'équipement de vie actuel du personnel navigant offre une protection prédéterminée contre les fortes accélérations, protection qui laisse à désirer. Les réactions physiologiques d'un pilote au stress causé par l'accélération dans des aéronefs tactiques sont typiquement non linéaires et elles varient selon la durée des accélérations pendant le vol, surtout la durée pendant laquelle l'aéronef se trouve en accélération négative Gz relative (c.-à-d. pendant l'effet « pousser-tirer »). Les progrès de la technique ont permis la mise au point de soupapes anti-G qui règlent et commandent le gonflage de la combinaison anti-G en temps réel pendant le vol, sur la foi de la durée pendant laquelle l'aéronef se trouve en accélération Gz. L'intégration des connaissances et de l'expérience d'experts ainsi que de la rétroaction biologique à un logiciel assurant le contrôle de la protection anti-G automatique en temps réel a fait l'objet d'études au début des travaux de l'ESI. En 2000, l'IMCME et ESI ont mené une étude expérimentale (protocole L-239) pour tester certaines des tables de protection non linéaire proposées pendant les transitions à faible intensité de l'effet « pousser-tirer ».

Dans le cadre de ce projet, ESI a intégré un modèle statistique évolué, ESIstat, pour étudier les données expérimentales recueillies lors de l'étude régie par le protocole L-239. Ce nouveau modèle statistique a été utilisé pour analyser et corroborer statistiquement un nouvel algorithme de commande de soupape anti-G de CF-18 visant à atténuer activement l'effet « pousser-tirer » grâce à la respiration à pression positive (PPB) pendant la phase « pousser ». Afin d'assurer la compatibilité avec les exigences des FC sur le compte rendu officiel des résultats d'essai menés sur des personnes, le modèle statistique mis en œuvre par ESI a par la suite été pourvu de capacité de production automatique de comptes rendus. En outre, comme partie intégrante du projet, les algorithmes de commande électronique en temps réel élaborés précédemment pour la soupape anti-G du CF-18 ont été partiellement modifiés pour assurer la compatibilité avec les spécifications du matériel et les protocoles d'essai utilisés pour les essais sur les personnes dans la centrifugeuse du laboratoire des forces de l'air des États-Unis.

On a démontré statistiquement dans des expériences sur table basculante que l'utilisation d'une respiration à pression positive de moyenne intensité (30 mm Hg) au cours d'une accélération négative -Gz, suivie du gonflage normal de la combinaison anti-G et d'une respiration à pression

positive d'intensité normale pendant la transition vers une accélération positive +Gz atténuait l'effet « pousser-tirer » et pouvait prévenir la diminution subséquente de la tolérance du pilote aux accélérations à cause d'un exposition aux accélérations négatives -Gz.

Contents

List of Figures III

List of Tables IV

1 Introduction 1

2 ESIstat Software Package 3

2.1 Framework of ESIstat 3

2.2 Preprocessing and Analysis of TSTAT-compatible data 9

2.3 Statistical Analysis 15

2.4 Graphics 19

2.5 Report Generation 22

3 Statistical Analysis of Tilt-Table Data 28

3.1 Design of Experiments 28

3.2 Subjects 29

3.3 Experimental Setup 29

3.4 Measurements 34

3.5 Data Analysis 34

3.6 Results: Overall Comparison 35

3.7 Results: Paired Comparisons 37

3.8 Discussion 39

References 45

Appendices 47

List of Figures

1 Overall Framework of Statistical Analysis of Experimental Data 6
2 Flow Chart of ESIstat Package 7
3 Flow Chart of TSTAT data analysis using ESIstat 8
4 Example of Systolic Blood Pressure and Mean Arterial Pressure on sample data. 16
5 Examples of Systolic Blood Pressure, Pulse Pressure and Stiffness Index on sample data. 17
6 Examples of ECG Peak Values, Heart Rate, and RR Intervals 18
7 Example of dynamic plot 23
8 Example of boxplots from the stacked dynamic data 24
9 Transition on the Tilt-Table. The tilt angle is first set at 15° (A1) for 15 seconds, then the subject is tilted 135° (A2) and kept there for another 15 seconds (A4). Then the subject is tilted back to head-up position and kept there for the rest of the run (A7). 31
10 Representative tilt-table run: Code name TPG-A27 (see Table 1 for description) 32
11 Subject on a motorized tilt-table. 33
12 Mean Values of Systolic Blood Pressure (SBP) for each of the Protocols (see Table 1 for the definition of codenames of the runs) 37
13 Ordered Mean Values of Systolic Blood Pressure (SBP) for each of the Protocols 38
14 Comparison of SBP in Controll-1 (tilt, no PPB) vs TPP-A2 (tilt, PPB in HDP all the way to HUP), p=0.01775 40
15 Comparison of DBP in Controll-1 vs TPP-A2, p=7x10^{-8} 41
16 Comparison of SBP in TPG-A77-1 (tilt, no PPB during tilt, PPB + G-suit applied reaching head-up) vs TPG-A27 (tilt, PPB in HDP all the way to HUP, G-suit applied upon reaching HUP). p=0.047 41
17 Comparison of DBP in TPG-A77-1 vs TPG-A27, p=0.72 42
18 Comparison of HR in Controll-1 vs TPP-A2, p=0.3113 42
19 Comparison of HR in TPG-A77-1 vs TPG-A27, p=0.986 43

List of Tables

1 Legend of Protocol L-239 Experimental Runs 30

2 Magnitude of BP Changes for Candidate Protections:
estimated during 5 seconds after head-up tilt is initiated 39

3 Magnitude of BP and HR Changes for Candidate Protections:
estimated from start of head down tilt till after 5 sec following onset of head-up tilt 40

1 Introduction

Modern fighter jets are far more technologically advanced and maneuverable than their World War II counterparts. Therefore, high G_z acceleration is more severe and is often encountered. In order to minimize the adverse physiological effects of G_z acceleration, an anti-G protection system is used to apply pressure to the pilot's body and provide positive pressure airflow to the pilot's breathing mask. Unfortunately, the technology of the currently used protection systems has remained conceptually similar to the original anti-G suit and valve of the 1940's [1]. Hence, current aircrew life support systems provide a pre-programmed sub-optimal level of high-acceleration protection.

The physiological responses of a pilot to the acceleration stresses seen in tactical aircraft will vary depending on the G-time history of the flight, especially the time at which the aircraft has spent in a relative negative Gz environment (i.e., Push-Pull effect). Clearly, the current protection scheme is often inadequate in protecting the modern fighter jet pilots from adverse physiological effects, such as insufficient eye-level blood pressure leading to potential G loss of consciousness (G-LOC). The shortcomings of the current protection system have prompted the need for new Anti-G valve hardware, and its accompanying protection profile generating software. Advances in technology have allowed for the development of a computer-controlled Anti-G valve that will adjust/control the Anti-G suit inflation in real-time based on the Gz-history of the aircraft. In our previous work, an electronic valve has been introduced [2]. However, software that is capable of generating nonlinear protection profile has not yet being made available. To alleviate this problem, the concepts of incorporating expert experience and knowledge as well as real-time biofeedback into software for automatic real-time Anti-G protection control has been developed at the previous stages of ESI's work.

In this project, ESI has integrated an advanced statistical modelling framework to investigate sets of experimental data collected in manned experimental studies (such as those using a centrifuge or tilt-table) performed using the Anti-G protection control architectures previously developed by ESI. The statistical framework (dubbed ESIstat) developed takes advantage of and is fully compatible with the set of advanced statistical modeling techniques (TSTAT Ver. 2) developed by DCIEM. This new statistical framework has been applied to a comprehensive set of physiological data collected by DCIEM and ESI in manned trials, and used to analyze and statistically substantiate a new CF-18 G-valve control algorithm for active attenuation of Push-Pull effect by applying mid-level Positive Pressure Breathing (PPB) during the Push phase.

To collect the data, an experimental study (Protocol L-239) has been carried out using the DCIEM tilt-table facility to investigate various aspects of subjects' physiological responses to candidate nonlinear G-suit and positive pressure breathing (PPB) protection schedules during low-intensity Push-Pull transitions. During the current phase of this project, the physiological data from the Protocol L-239 experiments has been formally investigated using a custom-designed suite of R statistical functions embedded into ESIstat. The purpose of the experiments and consecutive statistical investigation was to see if the Anti-G valve system would attenuate physiological effects typically associated with exposure to G_z forces, such as elevated blood pressure and a decrease in heart rate, which could potentially lead to G loss of consciousness. The data were fitted as *repeated measures* or *longitudinal* data to test whether the electronic anti-G valve system and candidate nonlinear protections had a significant effect in attenuating G_z tolerance.

The report consists of two parts. The first part of the report (Chapter 2) outlines the development of the ESIstat software package, along with examples, listings, helps, and justifications of the package. The second part (Chapter 3) presents in detail the statistical analysis results and procedures used to substantiate the new algorithm for active attenuation of Push-Pull effect.

2 ESIstat Software Package

The ESIstat software package consists of a suite of functions written in the R programming language [3] to perform statistical analysis on the Protocol L-239 Tilt-Table Experiments (see Section 3 "Statistical Analysis" for a detailed description of the Tilt-Table Experiments conducted at DCIEM). Protocol L-239 is attached in Appendix D. The ESIstat package was developed specifically to analyze data where the experimental subjects' observations are recorded over time. Such data are referred to as *repeated measures* or *longitudinal* data. ESIstat is fully compatible with R package TSTAT, custom-designed to handle centrifuge data packed in *jar* files with a defined layout. The capabilities of the ESIstat software package can be split into 4 main groups:

1. data preprocessing and analysis (for TSTAT-compatible data files),
2. statistical analysis,
3. graphics,
4. report generation.

This section will discuss in detail the specifics of the 4 main groups. The source codes for each of the ESIstat commands are provided in Appendix A, and a detailed description of each of the ESIstat commands, along with the description of the arguments, are provided in Appendix B.

2.1 Framework of ESIstat

This subsection present the overall framework and flow of logic that one is to follow in a typical working session with ESIstat. **Figure 1** presents a flow chart displaying the overall framework of the statistical analysis of the experimental data. First, the experimental results are stored in a data file, where they are compressed and stored into a series of Datatrac files using Java, after proper preprocessing using either LabView (used for the purposes of preprocessing in this project), Matlab, or C++ (see left-hand side of diagram).

Separately, a TSTAT object is created in R using the function **icreate** in the TSTAT package, which contains the name of the object, protection profiles, level of decimation, and the name of a datatrac file used to identify profiles for the *insertion* object. An insertion object is the object containing information relating to the data collected for one subject during one acceleration profile applied by a centrifuge or tilt-table. The Datatrac files are inserted into the TSTAT object using the TSTAT function **add**, after graphically selecting the *start* and *end* points for the profiles using the **ident** function. The **addDFC** function defines the channels and functions to apply for each protection profile, and the **getDFC** function is used to enter the TSTAT object into an R dataframe object. An R dataframe object can also be created using the TSTAT function **getd** and the ESIstat package for functions that cannot be called from within TSTAT (e.g., those that require more than one input channel or return more than one value). The subsequent statistical analysis is conducted using R statistical functions, including those available in ESIstat, and the results are saved in an Output Model, which can then be plotted or presented in a LaTeX document.

Figure 2 displays a flow chart of the ESIstat statistical package. The **Dfapplyfun** function loops through the insertions in the requested TSTAT object to apply specialized functions, as shown in the chart. The **Vintegratefull** function is used to perform an integration along a vector of data (the "full" refers to the fact that it can integrate below or above a given value). The **Gzpoints** and **Gzpoints2** functions locate points on a tilt-table profile. Currently, four points are located: start of transition from head-up (HU) to head-down (HD), time of reaching HD, start of transition from HD to HU, and time of reaching HU. The **calcDFC** function will apply an operation defined by the user (e.g. calculating means or minimums) to the data contained in the TSTAT object. The **verifyDFC** function is used to verify the data which may have been obtained directly by TSTAT, and the **recalcDFC** function is similar to the **calcDFC** function. The **Dfapplyfun** function outputs an R dataframe object, from which various statistical functions can be used (see Section 3 on "Statistical Analysis" as well as Appendix B for a more detailed description of these functions).

ESIstat is also capable of analyzing dynamic time-series data synchronized second-by-second. The **dynamicpts** function synchronizes the data from each insertion in the TSTAT object and interpolates the data to each second. The function accesses the "Whole" profile of TSTAT, and uses the start times and their corresponding time points to determine which insertion has the least amount of data points collected before the tilt is initiated. Similarly, an insertion with the least amount of data points recorded after the completion of a tilt is determined. The resulting dataframe consists of second-by-second data appearing in columns; the first row contains the time stamps, the second row contains the data for the reference channel. The user can plot the dynamic data points using the **dynplot** function for either all of the profiles or for specific profiles, or can calculate and plot the difference or delta values for the dynamic data points using the **dynplotdelta** function.

Figure 3 represents a Flow Chart of TSTAT compatible statistical data analysis using ESIstat. Using the ESIstat command **callTSTAT**, the user can use the TSTAT functions **select** to make the named insertion object the current insertion object, and the **getd** function to obtain the sampled data — for example, electrocardiogram data (ECG), systolic blood pressure (SYS), diastolic blood pressure (DIA), etc. Once the sampled data is obtained, there exists a number of ESIstat functions to convert the data into other useful data. For example, the **ecgp2hr** takes ECG data and generates heartbeat rate (HR) data. Similarly, the **ecgp2rr** generates RR intervals. The **sysdia2map**, **sysdia2pp**, and **sysdia2si** functions all take SYS and DIA data and generate mean arterial pressure (MAP), pulse pressure (PP) and stiffness index (SI), respectively.

ESIstat also provides a number of simple custom-designed functions of convenience for the user, such as the **last** function, which returns the last value in the data; the **calcslope** function, which calculates the best fit slope of the line between two vectors; the **firstquart** function, which returns the first quartile of a vector of data; and the **Vdecimate** function, which decimates a vector of data. See Appendix B for further details.

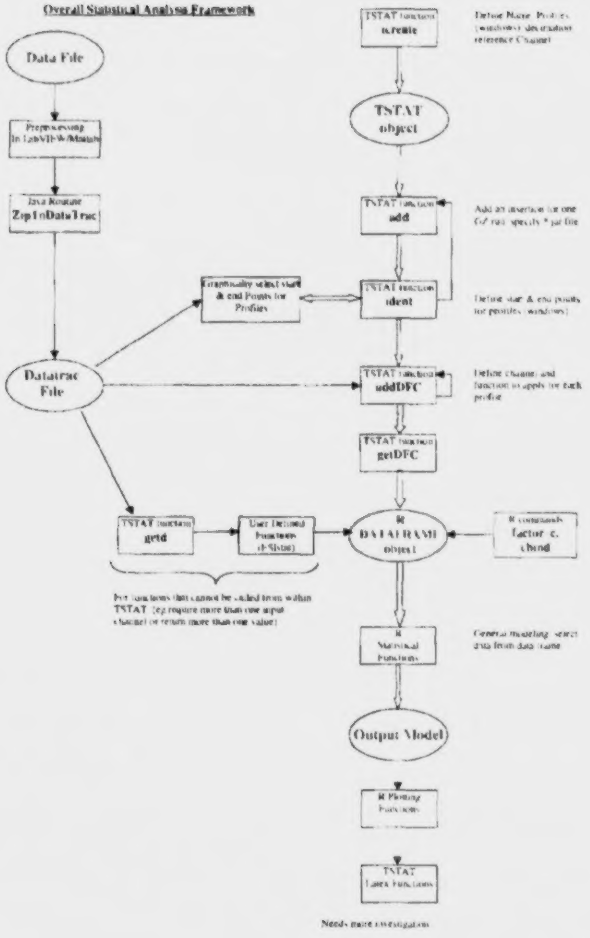

Overall Statistical Analysis Framework
Data File
Preprocessing In LabVIEW/Matlab
Java Routine ZipToDataTrac
Datatrac File
Graphically select start & end Points for Profiles
TSTAT function ts.create
Define Name, Profiles (windows), destination reference Channel
TSTAT object
TSTAT function add
Add an insertion for one GZ run; specify *.jar file
TSTAT function ident
Define start & end points for profiles (windows)
TSTAT function addDFC
Define channel and function to apply for each profile
TSTAT function getDFC
TSTAT function getd
User Defined Functions (S-Sixth)
R DATAFRAME object
R commands factor, c, cbind
For functions that cannot be called from within TSTAT (eg require more than one input channel or return more than one value)
R Statistical Functions
General modeling: select data from data frame
Output Model
R Plotting Functions
TSTAT Latex Functions
Needs more investigation

Figure 1: Overall Framework of Statistical Analysis of Experimental Data

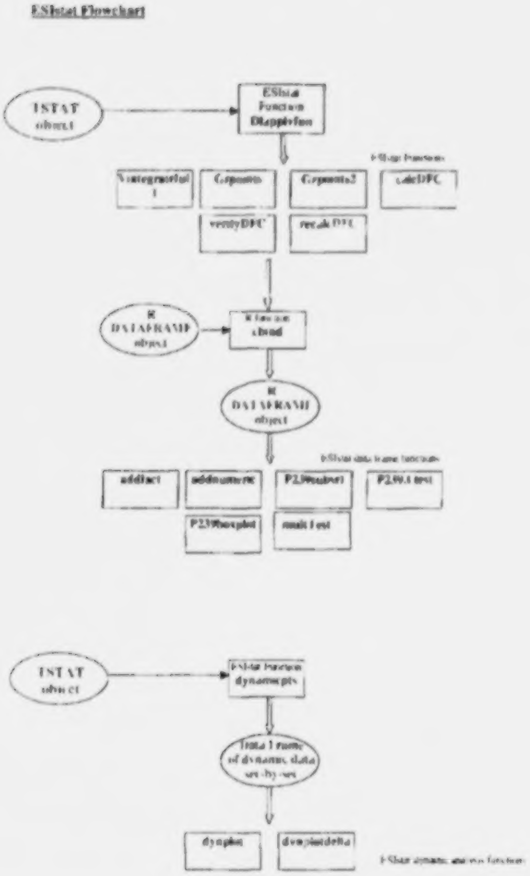

ESIstat Flowchart
ISTAT object
ESIstat Function
Gpoints
Gpoints2
calcDFC
verifyDFC
recalcDFC
R DATAFRAME object
R function cbind
R DATAFRAME object
addfact
addnumeric
P239boxplot
ISTAT object
ESIstat Function dynamicpts
Data Frame of dynamic data set-by-set
dynplot
dynplotdelta

Figure 2: Flow Chart of ESIstat Package

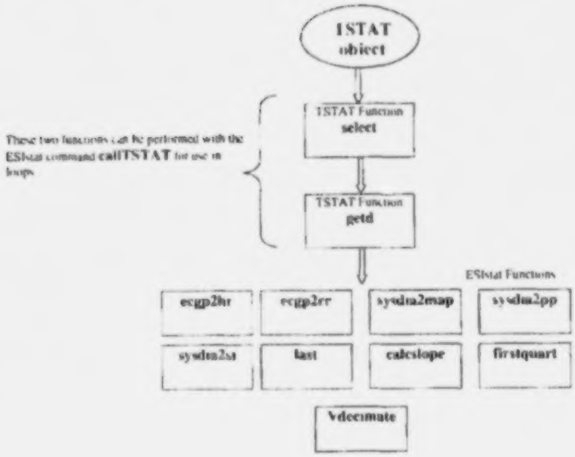

TSTAT object
TSTAT Function
select
TSTAT Function
getd
These two functions can be performed with the ESIstat command callTSTAT for use in loops
ESIstat Functions
ecgp2hr
ecgp2rr
sysdm2map
sysdm2pp
sysdm2st
last
calcslope
firstquart
vdecimate

Figure 3: Flow Chart of TSTAT data analysis using ESIstat

2.2 Preprocessing and Analysis of TSTAT-compatible data

ESIstat software, as discussed earlier, had been developed to work in conjunction with the TSTAT R package. The subset of ESIstat functions to be discussed in this subsection are intended to provide data analysis and preprocessing capabilities. These functions are applied to the data saved as TSTAT objects. The functions include:

(a) Dfapplyfun

(b) TSTATgetdata

(c) P239subset

(d) groupdata

(e) sysdia2map

(f) sysdia2pp

(g) sysdia2si

(h) ecgp2hr

(i) ecgp2rr

The **Dfapplyfun** function loops through the insertions in the TSTAT object (i.e. data from the tilt-table or centrifuge experiments stored in the format predefined by the TSTAT R function suite) and applies specialized functions to each insertion. For example, the specialized functions can be used for calculating mean blood pressure, heartbeat rate during a certain time period, etc. The **TSTATgetdata** function calls TSTAT to retrieve data. The **P239subset** function is a convenient function to extract a subset from a dataframe stored. The **groupdata** function is used to organize the various levels of factors in the object into one of two groups for comparison. Examples for the above functions are shown below (see also Tutorials in Appendix C for more detail):

```
> # Examples of the Dfapplyfun, P239subset and groupdata functions.
>
> # We wish to generate the delta values calculated as the difference between
> # post-tilt minimum and pre-tilt mean for systolic blood pressure (SBP),
> # which would be saved in the object "newdeltas".
>
> # Calculate the mean systolic blood pressure in the 15 seconds before tilt.
> newdeltas <- Dfapplyfun("calcDFC",cbind(fullnewGZpts[,1]-15,fullnewGZpts[,2]),
+ "newP239","Whole","HYS",1,207,"mean")
>
> newdeltas <- as.data.frame(newdeltas)
>
> # Calculate the minimum systolic blood pressure in the 5 seconds
> # starting at the HDT-HUT (head-down tilt-head-up tilt) transition.
> newtemp1 <- Dfapplyfun("calcDFC",cbind(fullnewGZpts[,3],fullnewGZpts[,3]+5),
+ "newP239","Whole","HYS",1,207,"min")
>
> # Here, note that the "newP239" refers to the TSTAT object containing all
> # of the runs in the experiment.
> # "Whole" refers to examining the whole profile of the experimental runs;
> # "HYS" refers to examining systolic blood pressure.
>
> # Calculate the delta
> newtemp2 <- newtemp1 - firstDF2[,1]
>
> # where "firstDF2" is a dataframe containing 5 sec pre-tilt final and 5 sec
> # post-tilt minimum values for systolic blood pressure (SBP),
> # diastolic blood pressure (DBP), Stiffness Index (SI),
> # and Heart Rate (HR), as well as their differences (deltas).
> # In this example, we are looking at just the SBP.
>
> # Combine the previous 3 columns of data into a dataframe and name the columns.
> newdeltas <- cbind(newdeltas,newtemp1,newtemp2,ExpRuns)
> names(newdeltas) <- c("SYSmeanpre","SYSminpost","DeltaSYS","ExpType")
>
> # Now we wish to extract a subset of this data frame, namely between
> # experimental runs with active PPB versus the relaxed control run.
> # We can do so using the "P239subset" command.
> PPBvsControl <- P239subset(newdeltas,4,c("Control1-1","TPP-A4-1",
+ "TPP-A4-2","TPP-A4-3"))
>
> # Now we may wish to display the summary of the ANOVA test.
> summary(aov(DeltaSYS ~ ExpType, data=PPBvsControl))
>
> # Finally, one may wish to group several types of experiments
> # together to strengthen the statistical power.  For example, up to
> # the HDT-HUT transition, experiments which include only PPB during
> # HDT are the same as experiments which include PPB during HDT and
> # G-suit after tilt.  If you are considering only the HDT data for
> # analysis, you may wish to group these experiments together.  This
> # can be done with the groupdata function:
>
> PPBvsNOPPB<-groupdata(SYSdyndata,"rownames",c("TPP-A4-1","TPP-A2",
+ "TPP-A4-2","TPP-A4-3","TPG-A47","TPG-A27"),
+ c("Control1-1","Control1-2","TGS-A7","TPG-A77-1","TPG-A77-2"),
+ c("PPBduringHDT","NoPPBduringHDT"))
```

```
>
> # Note that the second argument of groupdata could be either
> # "rownames" or a number that would represent the number of a column
> # to which grouping should be applied.
> # Note also that the experiments in the third and fourth arguments
> # will be classified as PPBdurngHDT and NoPPBduringHDT respectively.
> # Now an analysis could be performed to examine the effect of PPB
> # during HDT.
```

The **sysdia2map**, **sysdia2pp**, and **sysdia2si** functions all take a vector of systolic and a vector of diastolic blood pressure data as an input and return vectors of mean arterial pressure (MAP), pulse pressures (PP) and stiffness indices (SI), respectively. Similarly, the **ecgp2hr** and **ecgp2rr** all take a vector of electrocardiogram (ECG) peak values and converts them into heart rates (HR) and RR intervals, respectively. Examples for these functions are shown below, along with the corresponding plots (Figures 4-6) (see also Tutorials in Appendix C for more detail):

```
> # Examples of converting systolic and diastolic data to mean arterial pressure,
> # pulse pressure, and stiffness index.
>
> # We have a sample data frame called "mattsobj" that contains the 5 insertions
> # from a TSTAT object where tilt-table experimental data are saved.
>
> # First, we get the data for systolic and diastolic blood pressure (SYS and DIA)
>
> SYS<-TSTATgetdata("mattsobj","Whole","SYS")
> DIA<-TSTATgetdata("mattsobj","Whole","DIA")
> # The systolic and diastolic data were calculated beforehand and saved
> # in the datatrac file.
>
```

```
> # Now convert the systolic and diastolic data to mean arterial pressure (MAP).
> MAP <- sysdia2map(SYS,DIA)
>
> # Now do similar conversions to pulse pressure (PP) and stiffness index (SI).
> PP <- sysdia2pp(SYS,DIA)
> SI <- sysdia2si(SYS,DIA)
>
> # Plot the SYS, sending the plot to a Postscript file named "sysplot1.ps".
> postscript(file="PaperObjectWork/FinalReport/sysplot1.ps")
> par(mfrow=c(2,1))
> plot(SYS,type="l",ylim=c(50,150),xlim=c(0,70))
> title("Systolic Blood Pressure")
>
> # Now plot the MAP.
> plot(MAP,type="l",ylim=c(50,130),xlim=c(0,70),col="green")
> title("Mean Arterial Pressure")
>
> # This would end the "postscript" session and output the plot into the
> # file "sysplot.ps".
> dev.off()
null device
          1
>
> # Similarly, plot the SYS with the PP and SI.
> postscript(file="PaperObjectWork/FinalReport/sysplot2.ps")
> par(mfrow=c(3,1))
> plot(SYS,type="l",ylim=c(50,150),xlim=c(0,70))
> title("Systolic Blood Pressure")
```

```
>
> plot(PP,type="l",ylim=c(20,60),xlim=c(0,70),col="blue")
> title("Pulse Pressure")
>
> plot(SI,type="l",ylim=c(200,700),xlim=c(0,70),col="purple")
> title("Stiffness Index")
>
> dev.off()
null device
          1
>
> # Examples of converting electrocardiogram peak signals (ECGP) into
> # heart rate (HR) and RR interval.
>
> # First, we must get the ECGP (the ECG peaks were calculated beforehand and
> # recorded in the datatrac file.
> ECGP <- TSTATgetdata("mattsobj","Whole","ECGP")
>
> # Now, convert the ECGP into HR and RR
> HR <- ecgp2hr(ECGP)
> RR <- ecgp2rr(ECGP)
>
> # Now plot the data.
> postscript(file="PaperObjectWork/FinalReport/ecgplot.ps")
> par(mfrow=c(3,1))
> plot(ECGP,type="l",ylim=c(-1,1),xlim=c(0,70),col="red")
> title("ECG Peak Values")
>
```

```
> plot(HR,type="l",ylim=c(0,2),xlim=c(0,70),col="blue")
> title("Heart Rate")
>
> plot(RR,type="l",ylim=c(0,2),xlim=c(0,70),col="green")
> title("RR Interval")
>
> dev.off()
null device
          1
>
> # All plots generated are shown below.
```

2.3 Statistical Analysis

Among the capabilities of ESIstat are the ability to conduct statistical analysis on the repeated measures data that are generated from the tilt-table or centrifuge experiments. These functions are developed from existing R functions to analyze data stored in the format of the TSTAT objects. These functions include:

(a) P239.t.test

(b) multTest

The **P239.t.test** function performs a Standardized t test on two groups of data from a data frame, whereas the **multTest** function performs a multivariate Hotelling's T^2 test, the multivariant equivalent of the Standardized t test. Both functions return an object consisting of the calculated test statistic, degrees of freedom, the p-value as well as the 95% confidence intervals for the parameter of interest. The example of **multTest** function is shown below, whereas examples of **P239.t.test** function are given in Tutorial 2 of Appendix C:

```
> # Multivariate Analysis of Repeated Measures Data: Sample Analysis.
> # Test for differences between the systolic and diastolic blood pressure
```

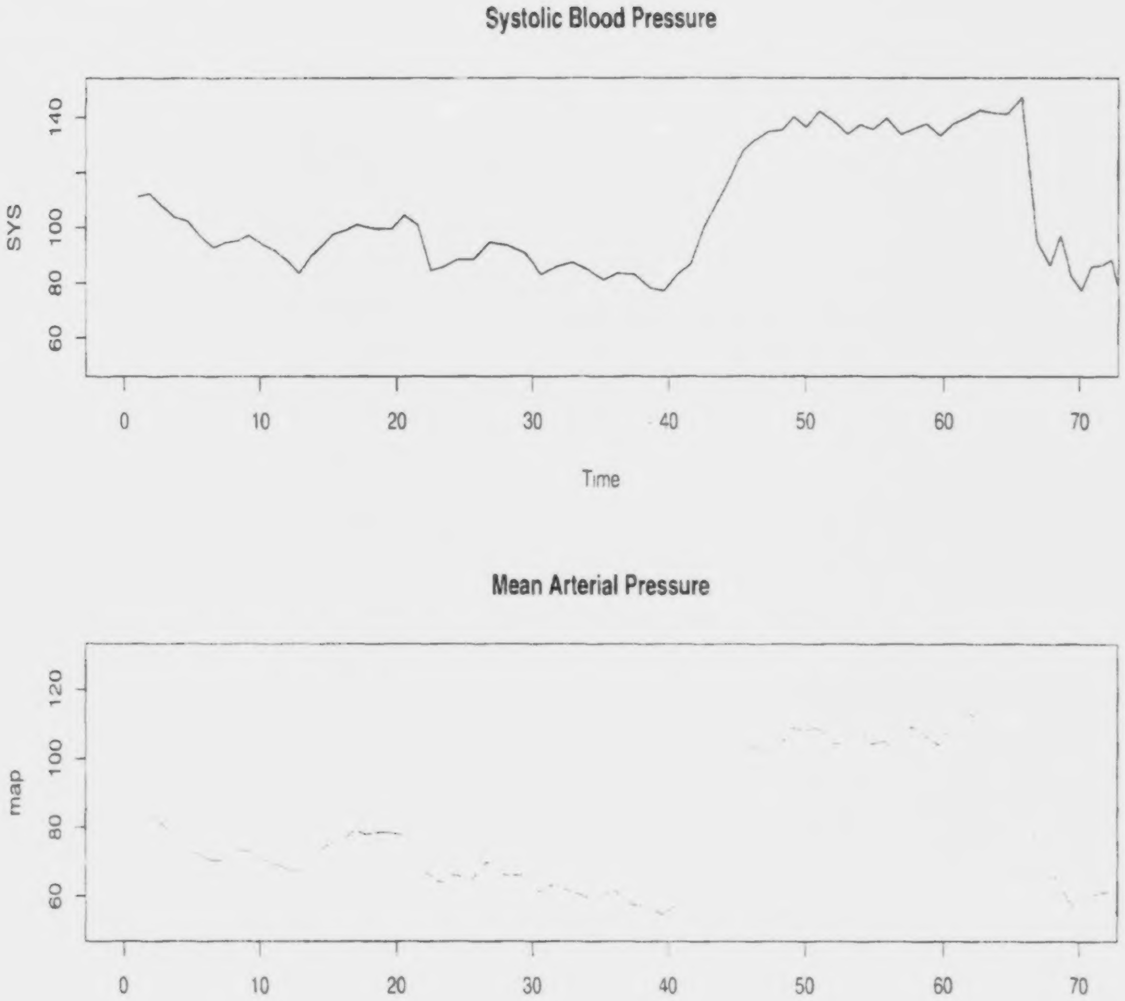


Figure 4: Example of Systolic Blood Pressure and Mean Arterial Pressure on sample data.

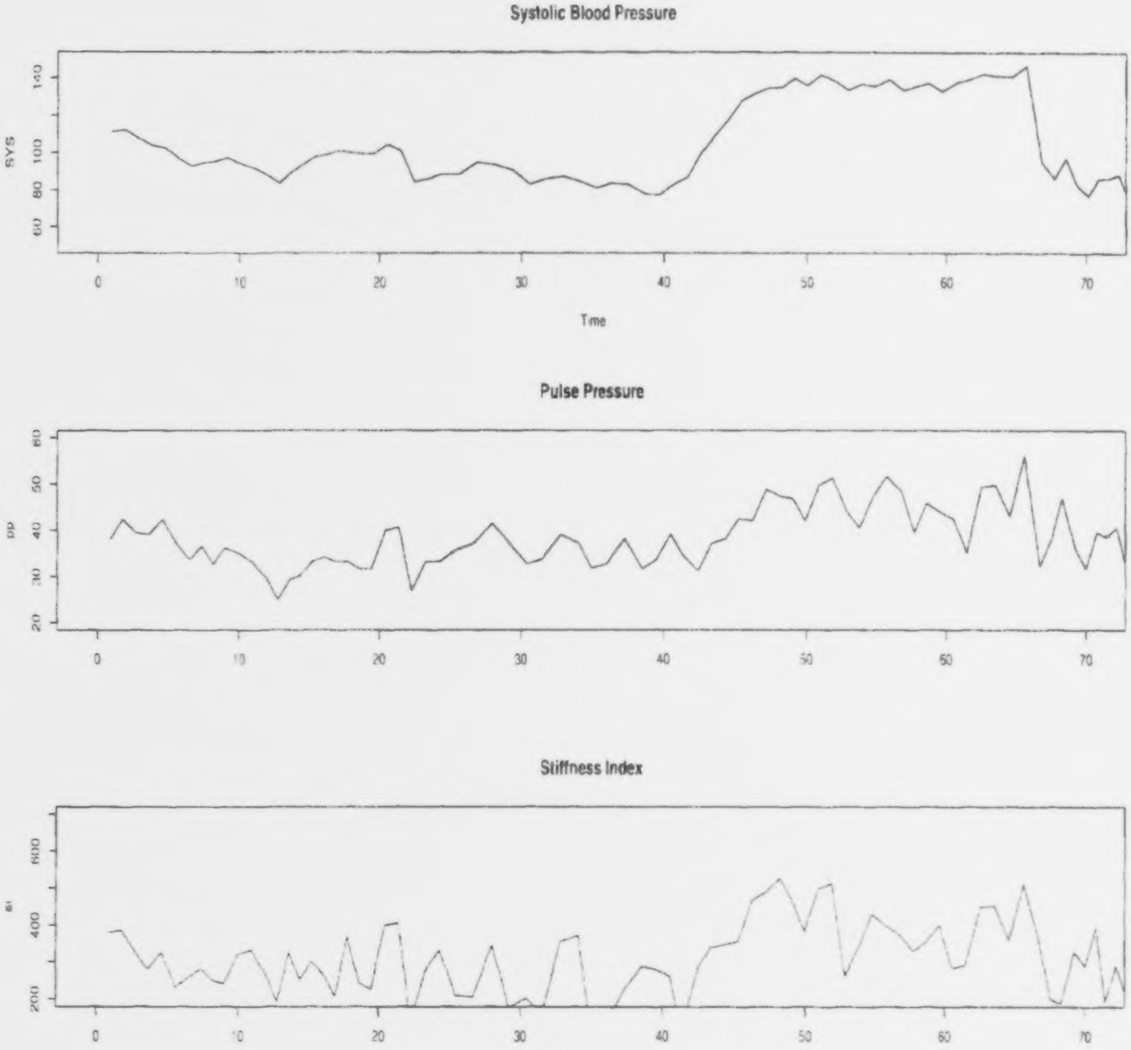


Figure 5: Examples of Systolic Blood Pressure, Pulse Pressure and Stiffness Index on sample data.

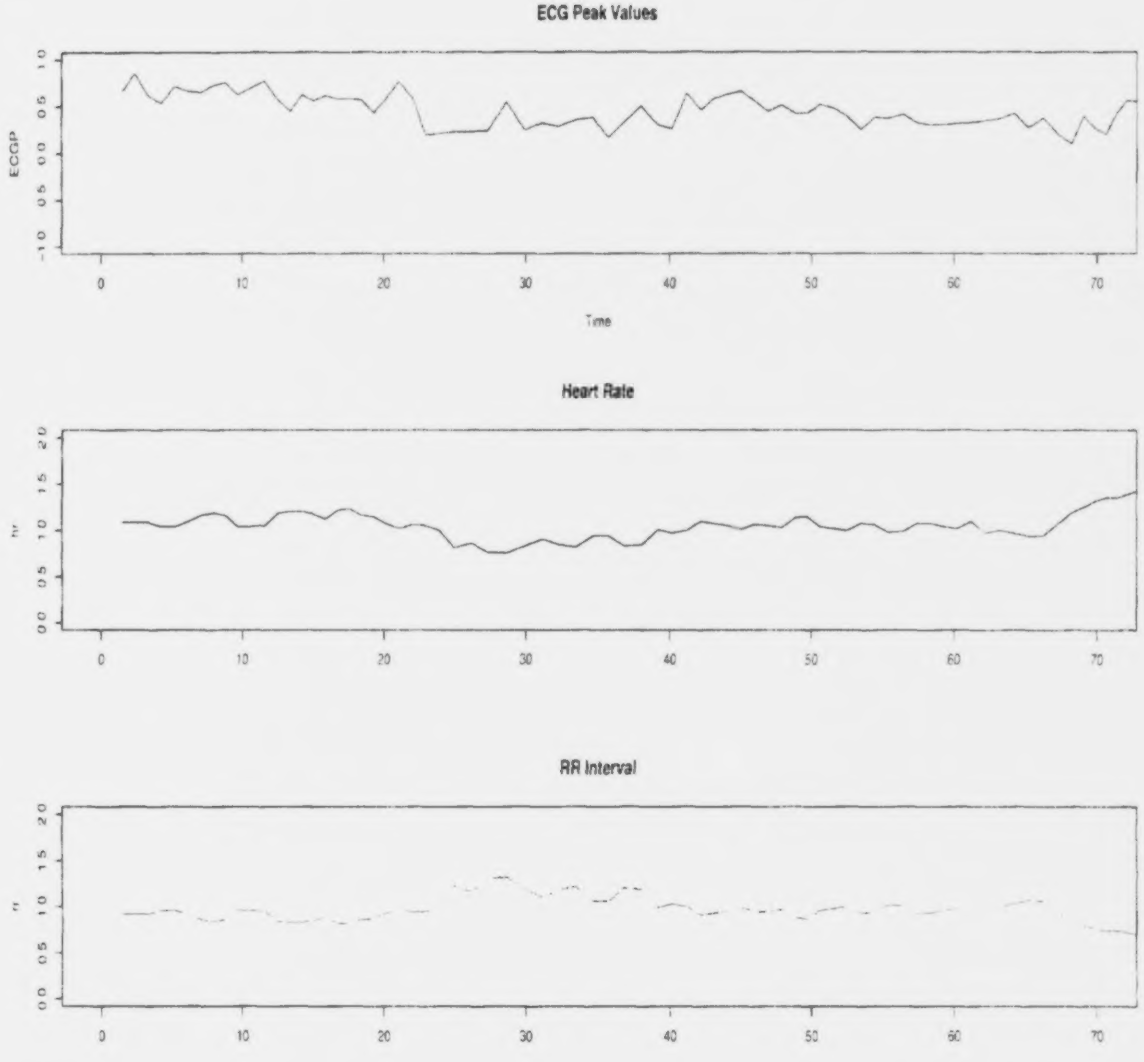


Figure 6: Examples of ECG Peak Values, Heart Rate, and RR Intervals

```
> # (SBP, DBP) for the following run.

> multTest(newdeltas,c(1,2),4,c("Control1-1","TPP-A2"))

The test statistic, pvalue and 95% confidence interval is:

[[1]]

NULL


$test.stat

         [,1]

[1,] 8.624015


$pvalue

           [,1]

[1,] 0.01719077


$conf.int

          [,1]     [,2]

[1,] -40.66717 15.06000

[2,] -57.58510 -6.16725
```

2.4 Graphics

The ESIstat software package also provides advanced graphical functions, including functions which are capable of generating both dynamic plots (i.e. time series plots of the tilt-table transitions) as well as static plots such as bar plots and box plots. The functions used extensively in the statistical analysis of the Tilt-Table experiments include:

(a) the **dynamicpts** command, which generates dynamic points generated from the TSTAT object containing the tilt-table or centrifuge experiments.

(b) the **dynplot** command, which plots one or more experimental runs of interest (see Appendix D for a description of the experimental runs used in the Tilt-Table study)

(c) the **dyndeltaplot** command, which calculates the delta values for the run types and plots them, and

(d) the **stackandbox** command, which rearranges a data frame of dynamic objects into a form suitable for boxplotting and/or analysis of variance.

Examples of the above functions are shown below, along with the dynamic plot and the boxplots (Figures 7-8) (see also Tutorials in Appendix C for more detail):

```
> # Dynamic graphics examples using ESIstat.
> # Dynamic data frames are very useful for visualizing the population-wide trend
> # for a given protection strategy.
> # By plotting various pairs of protection runs and noting how and when they
> # differ, one can determine where to best focus efforts on statistical analysis.
> # The dynamic data frames are calculated as follows.
>
> # Calculate the dynamic points from the data frames available.
> SYSdyndata <- dynamicpts("newP239",fullnewGZpts[,1:2],"HYS",plot=FALSE)
>
> # Note that the first column of the second argument is used for synchronization.
>
> # Assign proper rownames
> rownames(SYSdyndata) <- c("Gzpoints","Timestamps",ExpRuns)
>
> # Plot one of the runs of interest.
> postscript(file="PaperObjectWork/FinalReport/Cntrldyn.ps")
> dynplot(SYSdyndata,c("Control1-1"))
Hit <Return> to see next plot:
Cases are: Control1-1  ( red )
> dev.off()
```

```
null device
          1
>
>
> # Calculate the delta values based on the 5 seconds of rest before the tilt.
> SYSdyndelta <- dynplotdelta(SYSdyndata,5,c("Control1-1"))
>
> # Examining the rows of tilt data, it can be seen that columns 10 to 24
> # are the head-down tilt (HDT) period. Save this data.
> hdtSYSdyndata <- SYSdyndata[,10:24]
>
> # Now, to perform a t test based on this data, the columns must
> # be stacked one above another.
> # The "stackandbox" function returns a stacked data frame and displays a boxplot
> # of the data plotted against the values in the rownames of the data frame.
> postscript(file="PaperObjectWork/FinalReport/stacked.ps")
> stackedhdtSYSdyndata <- stackandbox(hdtSYSdyndata)
> dev.off()
null device
          1
>
> P239.t.test(stackedhdtSYSdyndata,1,2,c("Control1-1","TPP-A2"))

         Welch Two Sample t-test

data:  temp1 and temp2
t = -8.6317, df = 176.57, p-value = 3.455e-15
alternative hypothesis: true difference in means is not equal to 0
```

```
95 percent confidence interval:
 -28.63937 -17.98049
sample estimates:
mean of x mean of y
 102.0601  125.3700
```

2.5 Report Generation

In conjunction with the development of statistical, graphical and data-analytic functions for longitudinal data, as arising from the Tilt-Table experiments involving the CF18 Anti-G system, additional functions have been created to handle low-level LaTeX file manipulation. The functions developed include those for report generation. The report generation is approached in an object oriented manner, with a class of objects called "Paper Objects". These objects encapsulate private functions allowing low-level manipulations of LaTeX files and public functions which the user can call from outside the object. An R function called **CreatePaper** defines a new Paper Object.

The Paper Object comprises of sub-objects such as sections and subsections, which itself is composed of one of 3 object types: (i) paragraph or text, (ii) figures, and (iii) tables. The user has access to add, replace and remove sections or objects. When a Paper Object is defined using **CreatePaper** a LaTeX file with name specified by the user is generated, and all operations using the functions in the Paper Object are displayed onto the LaTeX file. If a file with the given name already exists, a message will appear stating so. The user can also specify the formatting of the documentation for the Paper Object (e.g. whether the object is to be formatted according to the IEEE reporting style, etc.). These public functions include the following:

(1) print

(2) get

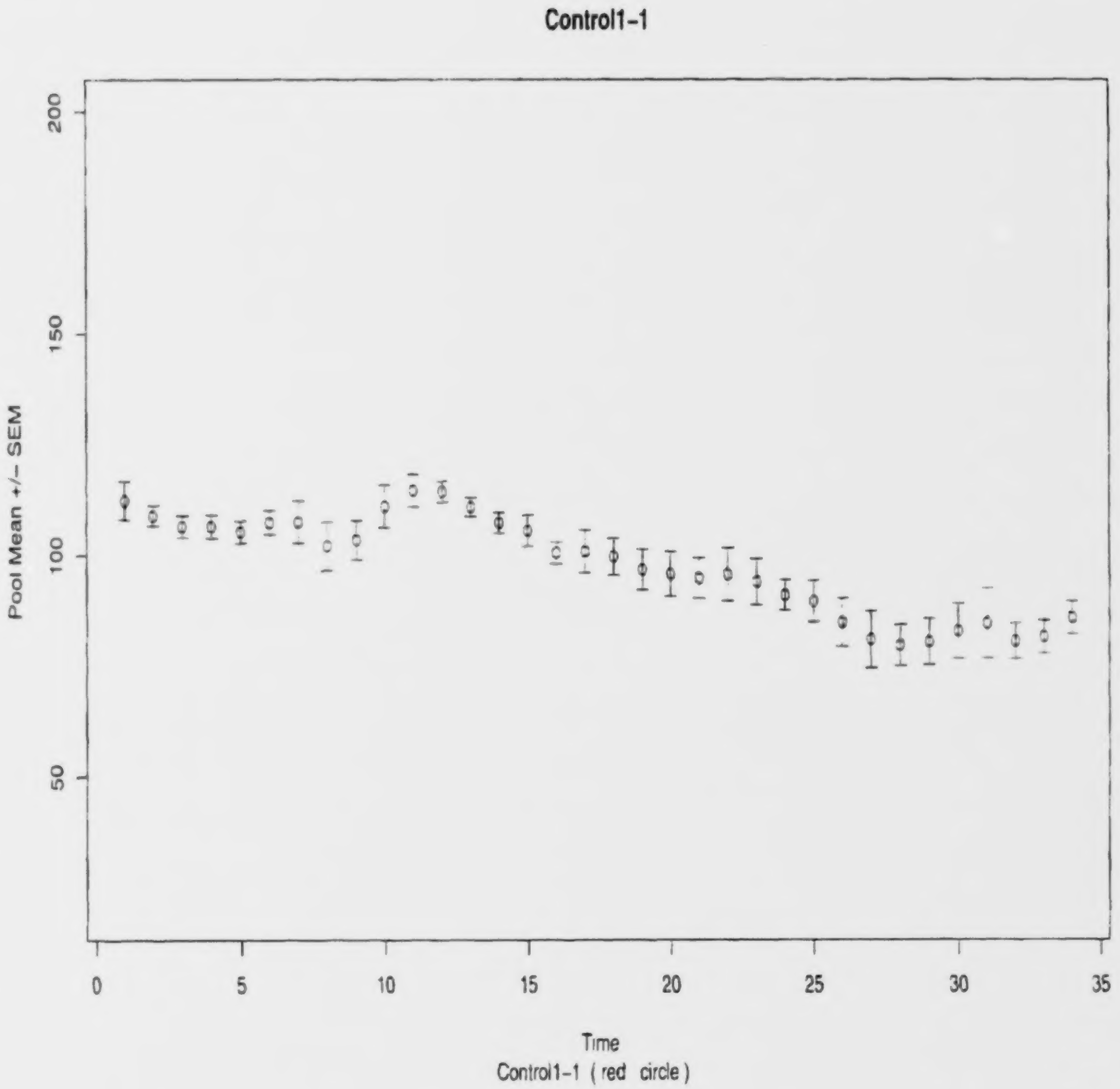


Figure 7: Example of dynamic plot

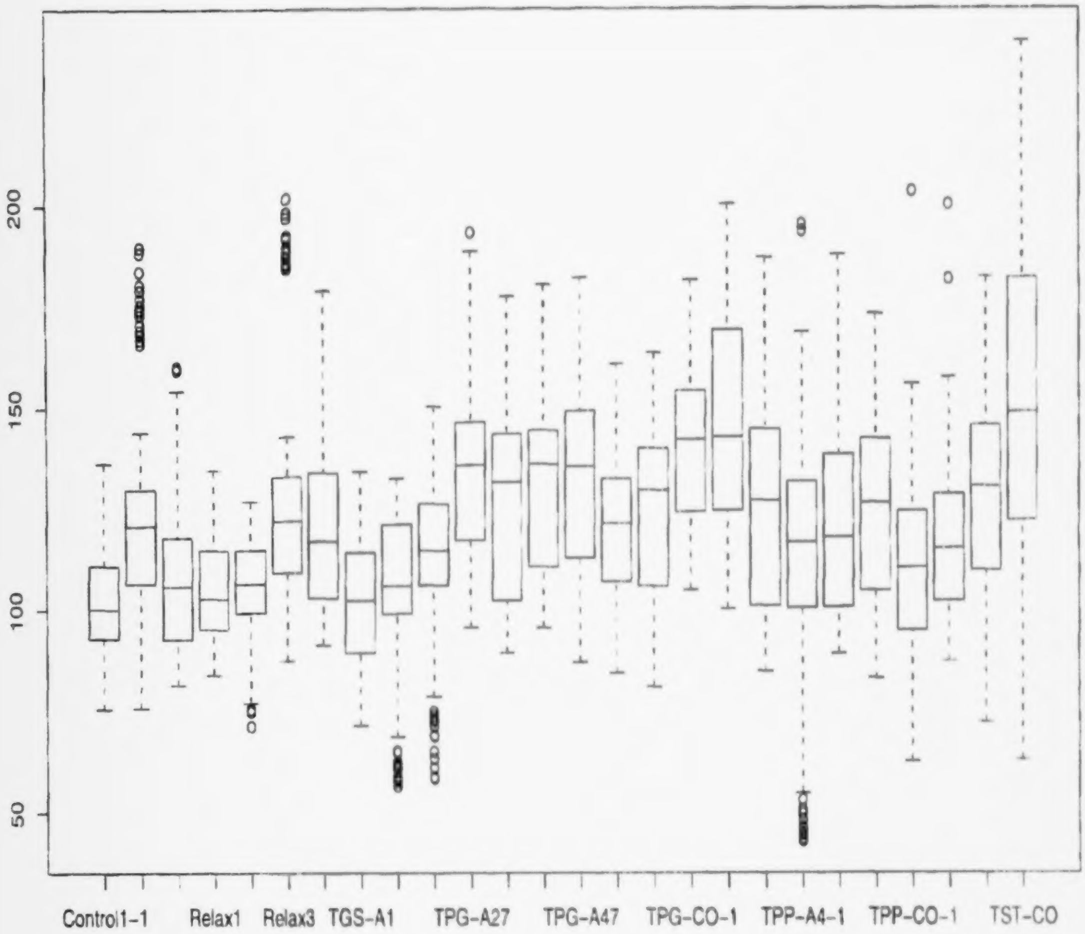


Figure 8: Example of boxplots from the stacked dynamic data

(3) addsection

(4) insertPar

(5) enddocument

The “print” function prints the description of the Paper Object, including the title, the file and type and added sections. The “get” function searches for a given section in the LaTeX file. The “addsection” function adds a section, with the user inputting the title, and the “insertPar” inserts text, figures or tables into the LaTeX file, with the user specifying the file name and type of file (“text”,“figure”,“table”). For example, if the user wishes to insert a figure into a given section, the user would input the following:

Object$insertPar(“*file name*”,“figure”)

The “enddocument” function ends the LaTeX document. Examples of the above functions are shown below:

```
> # Paper Object Software:
> # First, we wish to create a Paper Object, called "sample"
> sample <- CreatePaper("Test Title","testfile","IEEEsty")
Created File: /mnt/D/LinuxWork/Papers/testfile.tex
>
> sample

This is the print function for class PaperObj
Title:  Test Title
File: /mnt/D/LinuxWork/Papers/testfile.tex
Type:  IEEEsty
Sections: None

>
```

```
> # Now insert a section, called "Introduction"
> sample$addsection("Introduction")
This is the addsection function for class PaperObj
>
> sample

This is the print function for class PaperObj
Title:  Test Title
File:   /mnt/D/LinuxWork/Papers/testfile.tex
Type:IEEEsty
Sect1
    Title:  Introduction
    print.analysis method goes here


>
>
> # Now insert a paragraph here.
> sample$insertPar("/home/agalluz/PaperObjectWork/testing.tex","text")
>
> # Using this same function, a user can also insert figures or tables.
```

Some other functions defined in ESIstat for report generation include:

(a) **TexCurFig** – Send current figure to a LaTeX file,

(b) **TexStartTableFig** – Start a table of figures in a TeX file,

(c) **TexEndTableFig** - End a table of figures in a TeX file,

(d) **TexFindPSNum** - Find the next available Postscript file number for a given file name,

(e) **Texify** Compile a LaTeX file and send the output to the default DVI viewer,

(f) **TexRMAOV** - Send an Analysis of Variance (ANOVA) summary with more than one error stratum to a LaTeX file,

(g) **TexSimpleAOV** - Send an Analysis of Variance (ANOVA) summary with one error stratum to a LaTeX file,

(h) **TexSummaryAOV** - Send an Analysis of Variance (ANOVA) summary to a LaTeX file,

(i) **TexTable** - Send a table or matrix to a TeX file, and

(j) **TexTableFig** - Add the current figure to a table of figures in a TeX file, and

See Appendix B for a more detailed explanation of these functions.

3 Statistical Analysis of Tilt-Table Data

An operational flight envelope of a contemporary fighter jet may encompass cycles of $\pm G_z$ maneuvers. It is known that pilots' $+G_z$ tolerance may be significantly reduced when a $+G_z$ exposure has been preceded by a $-G_z$ period. This phenomenon is called the "Push-Pull" Effect. The intensity of negative G_z in aerial combat maneuvers can reach -2.0 to -2.5.

The current DCIEM Centrifuge can only generate positive G_z, whereas DCIEM Tilt Table generates only low-intensity Push-Pull ranging from at most -0.71 G_z to +0.866 G_z. Hence, one aim of the Protocol L-239 experiments (see Appendix D) was to test the utilization of suitable nonlinear combinations of G-suit pressure and Positive Pressure Breathing (PPB) profiles combined with head-down tilt (HDT) for inducing and exacerbating negative acceleration related physiological effects. It was expected that PPB combined with G-suit inflation in head-down position would cause certain physiological responses of a subject's cardiovascular functions that are similar to those resulting from an exposure to $-G_z$, such as:

(i) increase in venous return from lower body to upper body;

(ii) higher than normal blood pressure at head level:

(iii) decrease in heartbeat rate.

The other major goal of this study was to examine the efficiency of various G-suit and PPB nonlinear pressure schedules applied during both Push and Pull phases in protecting against the Push-Pull effect.

3.1 Design of Experiments

The experiments were performed on the DCIEM Tilt-Table with seated subjects wearing G-suit (STING) with CF PPB jerkin and PPB capable masks. It can be seen by reviewing Protocol L-239 that pre-defined profiles of Tilt-Table motion with different amplitudes and timings of G-suit pressures and PPB were applied. A pool of test subjects was recruited from DCIEM personnel medically qualified for PPB exposures.

All subjects were required to attend a training session (1 hour) on a separate day, prior to the actual experiments during which the subject was introduced to the experiment and the biomedical monitoring equipment. The subjects: (i) not to consume any alcoholic beverages with 24 hours of each series of experiments; (ii) to have a restful night before a test; (iii) not to perform heavy exercise on the day of a test; and (iv) to abstain from smoking, coffee, cola beverages and chocolate in the 3 hour period before a test, and a meal in the hour preceding it. A blood donation less than 30 days in advance of commencement of experimentation was defined as a contraindication to participation in the study.

All subjects were trained in performing PPB up to a maximum of 40 mmHg. The training included a comprehensive briefing on the test equipment, the experimental procedures and the physiological effects of PPB.

3.2 Subjects

8 volunteers (6 male, 2 female) between 18 and 50 years of age were recruited for this experiment. Each subject was provided with an information package which outlined the experimental protocol and which also contained the consent forms to be signed by the subjects. Participation was open to both civilian and military personnel. Subjects were screened for PPB exposure by a DCIEM physician. Investigations included ECG (electrocardiograph), CXR (Chest X-Ray), and detailed PFT (Pulmonary Function Test). Before the experiment subjects were monitored if there was any interim change in their medical status (Temperature $> 37°$, or Blood Pressure $> 140/90$). Subjects were recruited by personal contact.

3.3 Experimental Setup

The subjects were exposed to a series of 26 G_z transitions using the tilt-table. A typical transition sequence starts at a +0.866 G_z, then takes a subject to -0.71 G_z for about 15 seconds, and ends at +0.866 G_z. G-suit inflation pressures (0.4, and 5 psi) and PPB pressures (0,20,30, and 50 mmHg) were applied with computer-controlled Anti-G valve and PPB regulator (see Figure 9). Protection combinations included: no PPB and G-suit pressure, PPB pressure only, G-suit pressure only, and PPB and G-suit pressure. Table 1 displays each of the transitions along with their code names. Figure 10 shows an example of a transition used in the Protocol L-239 experiments, and Figure 11 shows a subject on a tilt-table.

Table 1: Legend of Protocol L-239 Experimental Runs

Test #	Test Code Name	Tilt	G-suit: When Tilt Active	G-suit Pres.	PPB: When Tilt Active	PPB: When G-suit Active	PPB Pres.
1	Relax1	No	No	0	No	No	0
2	Control1-1	Yes	No	0	No	No	0
3	Control2*	Yes	No	0	No	No	0
4	Relax2	No	No	0	No	No	0
5	TGS-CO	No	No	4	No	No	0
6	TGS-A7	Yes	After	4	No	No	0
7	TGS-A1	Yes	Before	4	No	No	0
8	TPP-CO-1	No	No	0	No	No	20
9	TPP-CO-2	No	No	0	No	No	30
10	TPP-A4-1	Yes	No	0	During	No	20
11	TPP-A4-2	Yes	No	0	During	No	30
12	TPP-A4-3	Yes	No	0	During	No	30
13	TPP-A2	Yes	No	0	During+ After	No	30
14	TPG-CO-1	No	No	4	No	During	30
15	TPG-CO-2	No	No	5.25	No	During	50
16	TPG-A77-1	Yes	After	4	After	During	30
17	TPG-A77-2	Yes	After	5.25	After	During	50
18	TPG-A47	Yes	After	4	During	After	30
19	TPG-A27	Yes	After	4	During+ After	Before+ During	30
20	TPG-A41-1	Yes	Before	4	During	After	20
21	TPG-A41-2	Yes	Before	4	During	After	30
22	TST-CO**	No	No	0	No	No	0
23	Relax3	No	No	0	No	No	0
24	TST-A1**	Yes	No	0	No	No	0
25	Control1-2	Yes	No	0	No	No	0
26	Relax4	No	No	0	No	No	0

* Head-down duration for this run was 40 sec.; for all others they were 15 sec.

** Straining had taken place in these runs.

Note:

1. G-suit Pressure was measured in psi.
2. PPB Pressure was measured in mmHg (4 psi = 206.8 mmHG, 5.25 psi = 271.4 mmHg).
3. HUT: 15°, HDT: 135°, Tilt Speed: 45°/sec.

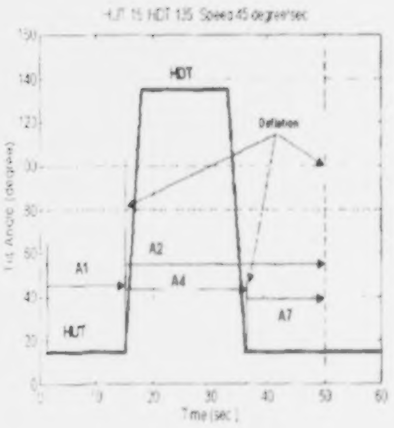


Figure 9: Transition on the Tilt-Table. The tilt angle is first set at 15° (A1) for 15 seconds, then the subject is tilted 135° (A2) and kept there for another 15 seconds (A4). Then the subject is tilted back to head-up position and kept there for the rest of the run (A7).

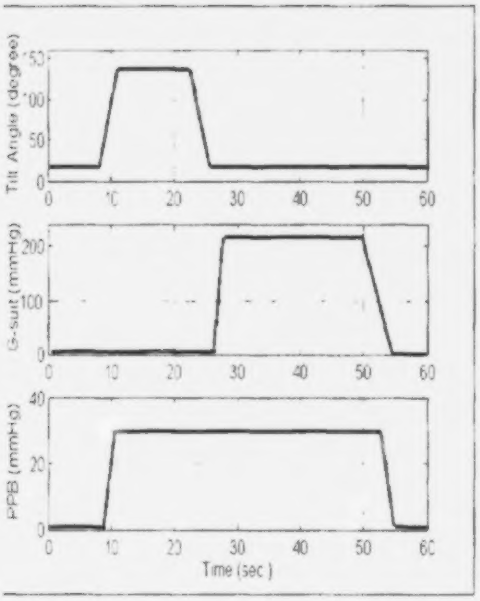


Figure 10: Representative tilt-table run: Code name TPG-A27 (see Table 1 for description)

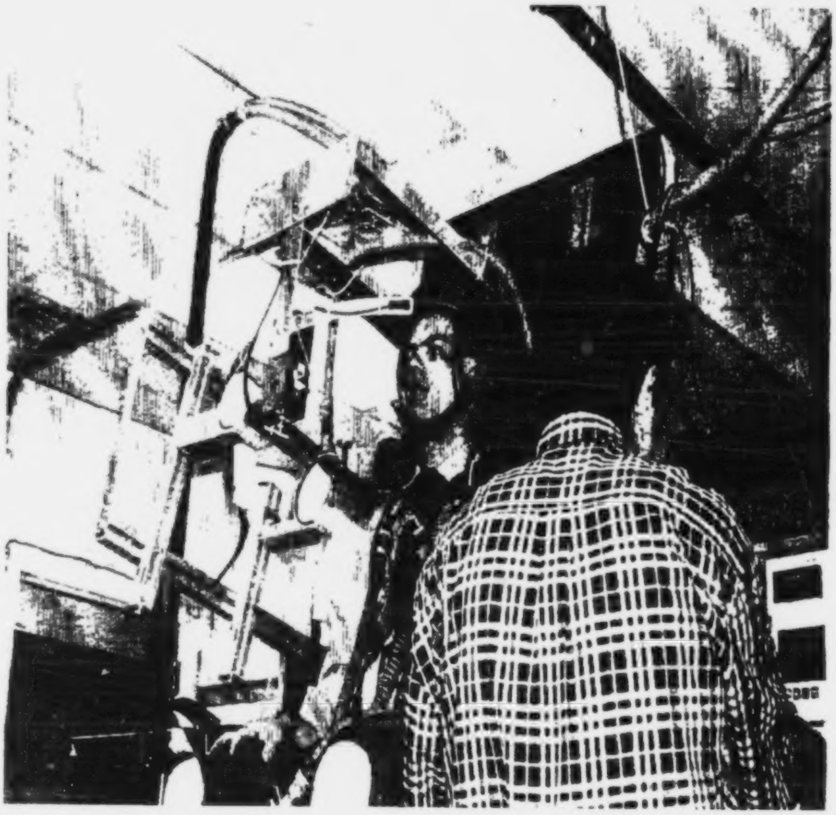


Figure 11: Subject on a motorized tilt-table.

3.4 Measurements

All experimental data were pre-processed to synchronize tilt-table runs using the onset of head-down tilt (HDT) as a reference. Next, all data were interpolated to obtain values of the physiological variables at one second intervals. The second-by-second systolic blood pressure (SBP), diastolic blood pressure (DBP), and heart rate (HR) data were then plotted to: (i) visualize the differences between various protection strategies during the push phase (HDT) and during the push-pull transition phase (beginning at the start of HDT to head-up-tilt (HUT) transition and lasting for 5 seconds); and (ii) detect and remove outliers.

3.5 Data Analysis

An analysis of variance (ANOVA) model for repeated measures was used to determine the significance of time-series SBP, DBP, and HR (the dependent variables) on the candidate treatments (nonlinear protection schedules) in each of the different recordings made where G-suit or PPB pressure were applied. A repeated measures ANOVA model can be expressed by the following equation [4]:

$$y_{ij} = \mu + \tau_i + \beta_j + \epsilon_{ij}, \quad i = 1, \ldots, a, \; j = 1, \ldots, b.$$

where y_{ij} is the observed data from the recordings of the SBP, DBP or HR, μ is the true theoretical mean of the data, τ_i is the effect of the i_{th} treatment (here it would be the specific protocols of G-suit or PPB pressure applied), β_j is a parameter associated with the j_{th} subject, the ϵ_{ij} is the error term, and a and b refer respectively to the number of treatments and to the number of subjects. We assume that the treatments are fixed (so $\sum_{i=1}^{a} \tau_i = 0$), and that the subjects used are a random sample of subjects from a larger population of potential subjects.

An univariate paired t-test was used to determine if there were any significant differences in blood pressure and heart rate between the different tilt positions [5]. A multivariate t-test was also used to test for significant differences in a combination of systolic and diastolic pressures between the tilt positions [6]. A significance level of $p < 0.05$ was used for all statistical tests [7]. See the ESIstat Software definition is Section 2 for a detailed explanation of the software used to analyze the data.

3.6 Results: Overall Comparison

In general, Push-Pull maneuvers cause significant drops in heart-level SBP and HR during -Gz (push) phase. Based on the previous analysis of CAPS data [8,9] and recent tilt-table studies [10], it has been found that these drops may temporarily worsen during the transition from -Gz to +Gz, which could drive head-level BP below the acceptable threshold level and lead to G-LOC.

To perform a statistical analysis on any given signal from an experiment the first step is to decide on the desired indices. In other words, what aspect of the signal is of interest? Specific indices could be the mean of the signal, the maximum of the signal during a given time interval, the standard deviation of the signal, etc. In the preliminary investigation performed, two main approaches were used in analyzing Protocol L-239 experiments. Both approaches focused on the time regions including the pre-tilt resting value of the signal, and the post-tilt values since. The pre-tilt values are used as a baseline while the post-tilt values indicate the effectiveness of the protection strategy. The two approaches are as follows:

1. For each physiological variable (eg. SBP, DBP, Heart Rate) the LAST GOOD value in the 5 seconds before tilt was selected. It was reasoned that the LAST value would best indicate the physiological state of the subject before tilting. For the post-tilt region the MINIMUM value in the 5 seconds starting at HDT-HUT transition was selected. It was found by experience that at least 5 seconds post-tilt is required to discern between protection effects. This is the strategy used to generate firstDF2. The graphical exploration of this data frame was handed out at meetings with the Scientific Authority and appears on PCESI computer in the file (see Tutorial 2 in Appendix C for more detail):

   `"d:\linuxWork\p239results\FirstTry.zip"`

2. For each physiological variable (eg. SBP, DBP, Heart Rate) the MEAN value in the 15 seconds before tilt was selected. Here it was reasoned that selecting only one value (ie

the LAST value) results in too much noise to generate reliable results. It was expected that a MEAN over 15 seconds would yield more consistent results. For the post-tilt region the MEAN value in the 5 seconds starting at HDT-HUT transition was selected. Again, it was reasoned that the MEAN post-tilt value would yield consistent results. This is the strategy used to generate firstDFmeans. The graphical exploration of this data frame was handed out at meetings with the Scientific Authority and appears on PCESI computer in the file (see Tutorial 2 in Appendix C for more detail):

`"d:\linuxWork\p239results\SecondTry.zip"`

In the end, as the following presentation will show, it has been found that a combination of the two aforementioned approaches would yield the best results. The pre-tilt MEAN value (in the 15 seconds before tilt) yields consistent baseline values and the post-tilt MINIMUM value (MIN in the 5 seconds starting at HDT-HUT transition) best displays the effectiveness of the control strategy. To summarize the data, we take the difference (delta) between the pre MEAN and post MAXIMUM values. The result is stored in *newdeltas* data frame. Note that to calculate the MINIMUM heart rate, the MAXIMUM RR interval must be used. The heart rate values appear in hrDF dataframe. To summarize, to quantify the effects of candidate nonlinear protection strategies proposed in Protocol L-239 on the drops, relative changes in blood pressures(delta SBP and DBP) and changes in heart rates (delta HR) were calculated as the difference between the mean value during 15 seconds prior to the onset of a head-down tilt and minimum value during the transition from a subject's head-down position to head-up position.

Figures 12 and 13 below show the barplots of the pooled means of delta SBP for subjects exposed to the experimental runs described earlier. Figure 12 shows the pooled means of delta SBP in order of performance of the experimental runs conducted on each subject, while Figure 13 shows the same pooled means ordered numerically. It should be noted that the first 6 runs in Figure 13 did not involve tilt transitions (3^{rd} and 6^{th} runs involve only inflation of G-suit and PPB, respectively). For the control runs that involve tilt transition only, no PPB and G-suit pressures was applied. The second control run (Control1-2) has smaller

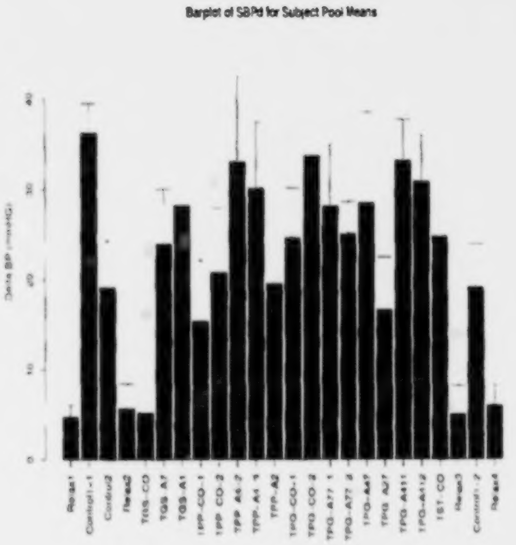


Figure 12: Mean Values of Systolic Blood Pressure (SBP) for each of the Protocols (see Table 1 for the definition of codenames of the runs)

dynamic variability of SBP than that in the first one (Control1-1). This is possibly due to a habituation effect noticeable among the subjects, where after repeated experimental runs, the subjects' bodies adapt to the rigors of the experimental conditions. Figure 13 shows that the experimental run TPG-A27 (refer to Appendix D for PPB and G-suit pressure profiles) yields the smallest drop in SBP in the runs that involve tilt transitions.

Repeated measures ANOVA analyses applied to the above data sets showed a significant difference in delta SBP (p=0.03135), delta DBP (p=0.0002) and delta HR (p=0.0036) among different runs with experimental protection schedules. Further evaluations using paired t-test have been conducted to find source of the differences. The results of these evaluations are summarized in subsection *Paired Comparisons* below.

3.7 Results: Paired Comparisons

Let us investigate whether the PPB applied during $-G_z$ phase has any effect on mitigating the physiological responses during $+/-/+G_z$ transitions. We take those runs where PPB was not activated during tilt phase as reference, and compare them with other runs where PPB

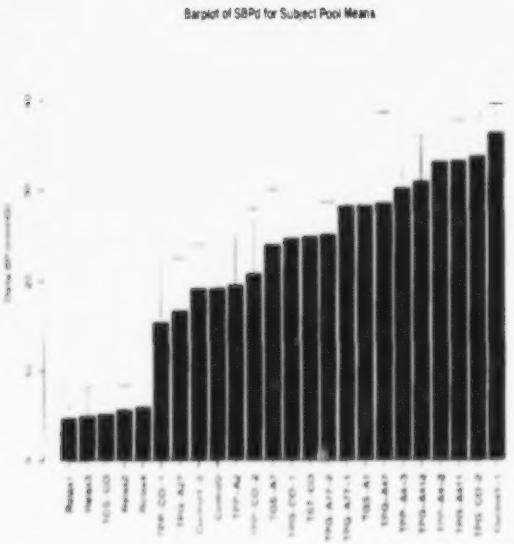


Figure 13: Ordered Mean Values of Systolic Blood Pressure (SBP) for each of the Protocols

was activated during tilt phase. Particularly, we performed paired t-test on two pairs of runs: Control1-1 versus TPP-A2, and TPG-A77-1 versus TPG-A27. In the first pair, Control1-1 is a reference run where no PPB and G-suit pressure was applied. In the candidate run TPP-A2, PPB was applied during tilt and transition phases. In the second pair, TPG-A77-1 is a reference run where PPB and G-suit pressure were applied immediately after the tilt transition. This profile represents the standard Anti-G protection schedule currently in practice. We should note that we do not compare TPG-A77-2 with TPG-A27 since in TPG-A77-2 the G-suit pressure was set at 5.25 psi, whereas the G-suit pressure for TPG-A77-1 and TPG-A27 were set at 4 psi.

Figures 14 and 15 show the blood pressure (SBP and DBP) and tilt-table profiles in the first paired runs. Figures 16 and 17 show the results for second paired runs. Table 2 summarizes the comparison results on the difference in delta SBP, delta DBP and delta HR that are all estimated for the time period of 5 seconds following the onset of head up tilt. It shows that PPB applied during $-G_z$ phase reduces the drop by about 8 to 20 mmHg in SBP during the transition phase (p=0.018). Similar results are obtained for DBP

Table 2: Magnitude of BP Changes for Candidate Protections:
estimated during 5 seconds after head-up tilt is initiated

	Controll-1 vs TPP-A2	TPG-A77-1 vs TPG-A27
Δ SBP (mmHg)	$14.37 \pm 5.80^{(*)}$	$7.82 \pm 7.81^{(**)}$
Δ DBP (mmHg)	$10.86 \pm 3.88^{(***)}$	NS
Δ HR	NS	NS

(*) SBP: $p < 0.018$
(**) SBP: $p < 0.047$
(***) DBP: $p < 0.0001$
NS: Not Significant

in the comparison of Controll-1 versus TPP-A2, where PPB applied during $-G_z$ phase reduces the drop by about 11 mmHg in DBP during the transition phase ($p=7\text{x}10^{-8}$). In the comparison of TPG-A77-1 versus TPG-A27, it has been established that if the standard protection schedule is augmented with PPB applied during $-G_z$ phase, then the drop is SBP during the transition phase can be further reduced by up to 15 mmHg ($p=0.047$). However, in the comparison of TPG-A77-1 versus TPG-A27, there is no significant difference in DBP ($p=0.72$). In both of the previous comparisons, there is no significant change in delta HR ($p=0.311$ and $p=0.986$, respectively; see also Figures 18-19 below).

Table 3 summarizes the comparison results on the difference in delta SBP, delta DBP and delta HR that are estimated during the time period that lasts from the moment when the head down tilt is initiated until the time moment defined as 5 seconds after the onset of head up tilt is initiated. The results in Table 3 further substantiate the potential of PPB applied during the $-G_z$ phase as a potential countermeasure that attenuates Push-Pull effects.

3.8 Discussion

The tilt-table experiments performed on relaxed unprotect subjects as a part of Protocol L-239 study show the adverse physiological effects resulting, as indicated in [8] and [9], from exposure to -G_z: reduced blood pressure and heartbeat rate. The application of PPB during -G_z followed by the addition of G-suit pressure right on the onset of Push-Pull transition attenuates the Push-Pull effect and may prevent the subsequent decrease in G_z-tolerance due

Table 3: Magnitude of BP and HR Changes for Candidate Protections: estimated from start of head down tilt till after 5 sec following onset of head-up tilt

	Controll-1 vs TPP-A2(*)	TPG-A77-1 vs TPG-A27(**)
Δ SBP (mmHg)	10.08 ± 3.24	10.97 ± 3.65
Δ DBP (mmHg)	6.61 ± 2.13	9.02 ± 2.93
Δ HR	NS	NS

(*) SBP: $p < 0.0001$, DBP: $p < 0.0001$
(**) SBP: $p < 0.0001$, DBP: $p < 0.0001$
NS: Not Significant

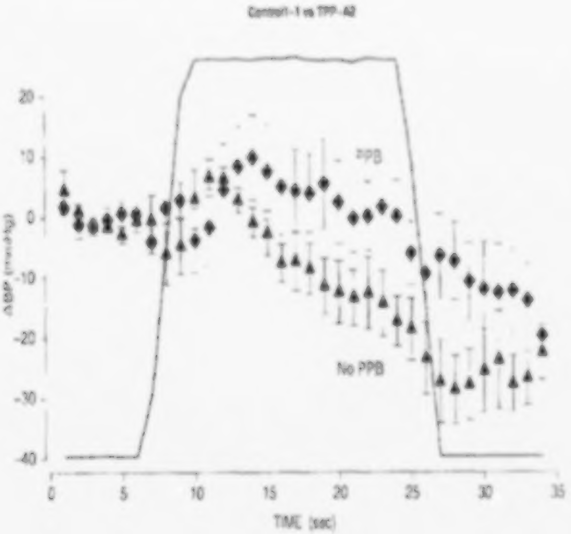


Figure 14: Comparison of SBP in Controll-1 (tilt, no PPB) vs TPP-A2 (tilt, PPB in HDP all the way to HUP), p=0.01775

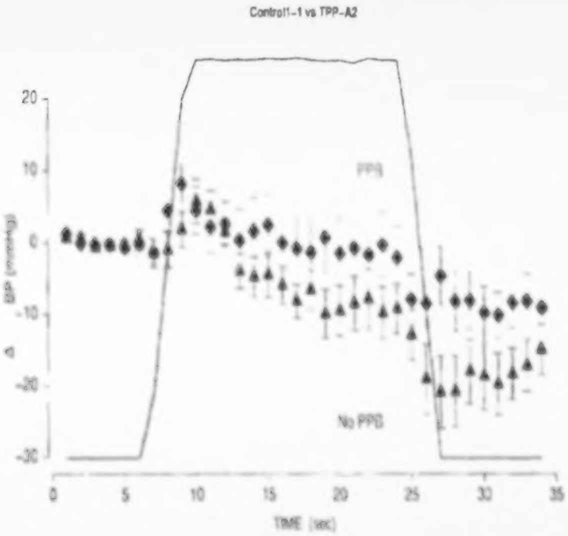


Figure 15: Comparison of DBP in Control1-1 vs TPP-A2, $p=7x10^{-8}$

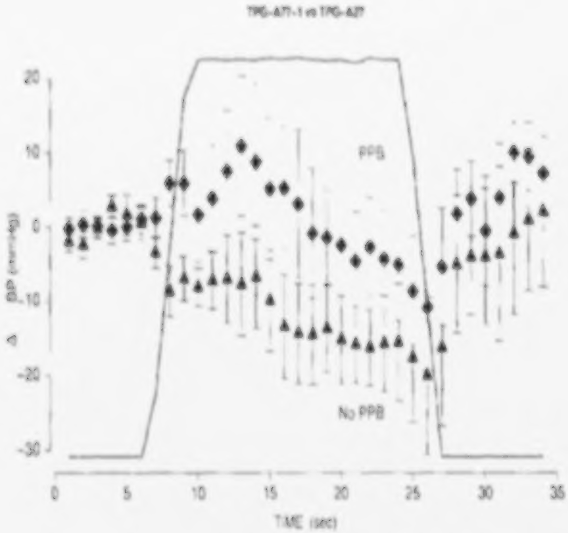


Figure 16: Comparison of SBP in TPG-A77-1 (tilt, no PPB during tilt, PPB + G-suit applied reaching head-up) vs TPG-A27 (tilt, PPB in HDP all the way to HUP, G-suit applied upon reaching HUP), p=0.047

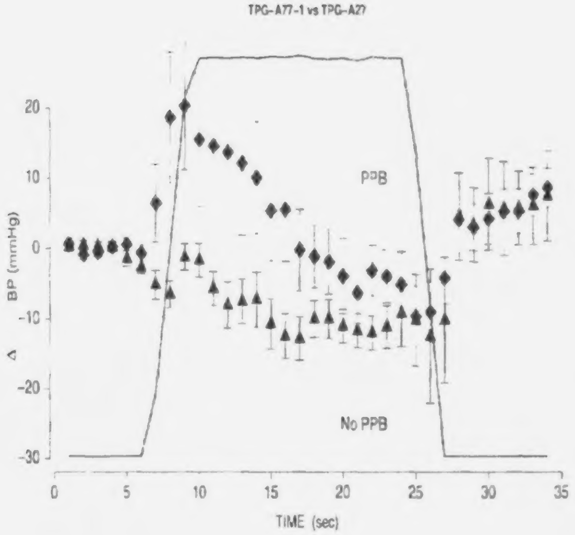


Figure 17: Comparison of DBP in TPG-A77-1 vs TPG-A27, p=0.72

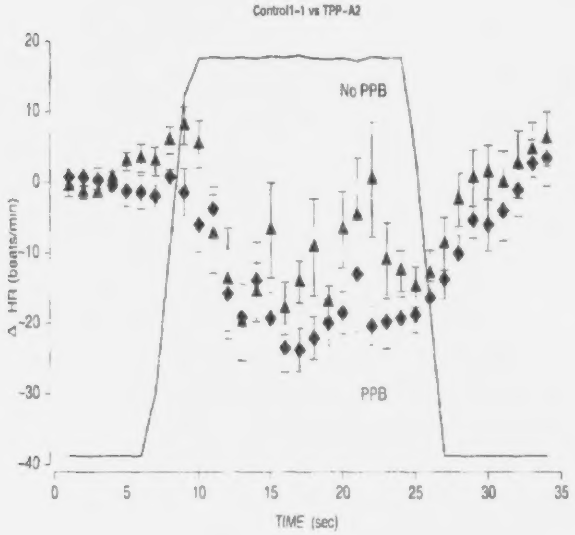


Figure 18: Comparison of HR in Controll-1 vs TPP-A2, p=0.3113

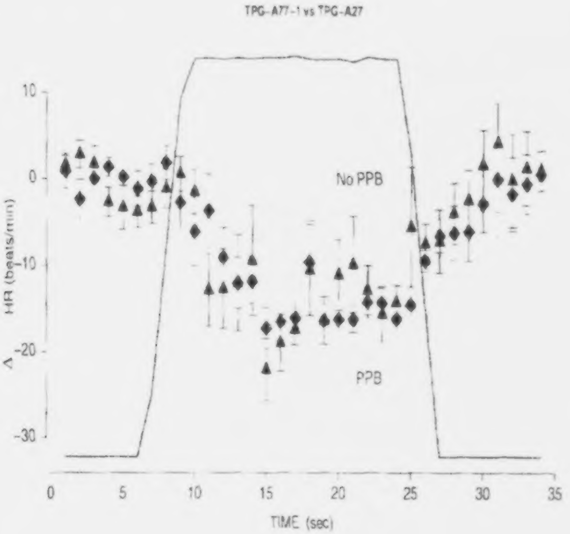


Figure 19: Comparison of HR in TPG-A77-1 vs TPG-A27, p=0.986

to -G_z exposure. It is demonstrated by the statistical analysis of the experimental results that the drop in blood pressure is reduced during transition phase.

It is noted from the comparison of TPG-A77-1 versus TPG-A27 (refer to Figure 16) that for the same levels of G-suit and PPB pressures applied during +G_z phase, a greater increase in systolic pressure can be obtained when PPB is used during the Push phase. This implies that as a direct benefit of applying PPB during -G_z phase in Push-Pull maneuvers, a subject may be able to sustain a higher intensity +G_z without experiencing adverse physiological effects.

It has been observed (refer to Figures 16 and 17) that the difference in diastolic pressure due to the application of PBB gradually diminished during the Push phase, yet the difference in systolic pressure is maintained. This might indicate that PPB applied during -G_z induces high levels of vigorous heart work, thereby increasing the pulse pressure.

The timing of the PPB application during -G_z phase might prove to be a critical factor in establishing the maximum effectiveness of this candidate Push-Pull protection. It can be seen from Figure 16 that heart-level SBP increases only during the initial few seconds after the application of PPB during the head-down phase. Afterwards SBP gradually decreases. Whether PPB has to be initiated on the onset of -G_z, or right after -G_z has been reached remains to be investigated. From the practical point of view, however, it seems that the approach of initiating PPB right on the onset of -G_z is the practical way to go, and this is because the actual duration of -G_z phase in an actual flight is not know a priori and can be quite short. At the same time, short durations of -G_z phase might degrade a subject's Push-Pull tolerance even more than long durations. This is due to the fact (see CAP's data [8] and Figures 14 and 18) that the recovery in blood pressure and heartbeat rate drops gets established only after a subject has spent a ceratn amount of time in -G_z environment. Hence, it seems at the moment that initiating PPB right on the onset of -G_z is both practical and physiologically advantageous.

Further studies in a push-pull capable centrifuge are required to examine the protection performance and to optimize the PPB and G-suit inflation schedules. These studies are planned on such a centrifuge that is currently being assembled at DCIEM.

References

[1] Wood, Earl H., Edward H. Lambert, Edward J. Baldes, and Charles F. Code., "Effects of Accleration in Relation to Aviation", *Federation Proceedings*, Vol. 5, 1946.

[2] Goldenberg, A.A., D.F. James, P.S.E. Farrell, D. Meidan, and Y. Shapiro., "A New Method for Anti-G Pilot Suit Pressure Control in High Energy Maneuvering Aircraft, Phase 4: Integration of Positive Pressure Breathing and Study of a Multi-Bladder Physiological Protection System", *final report* 1992.

[3] **An Introduction to R**, version 1.0.1, R Development Core Team, April 14, 2000.

[4] Montgomery, D.C., **Design and Analysis of Experiments**, 4th ed., 1997.

[5] **S-PLUS Guide to Statistical and Mathematical Analysis**, version 3.2, StatSci, Mathsoft Inc., December 1993.

[6] Srivastava, M.S., **Methods of Multivariate Statistics**, 1999, an unpublished revision of Srivastava, M.S. and E. Carter, **An Introduction to Applied Multivariate Statistics**, North Holland 1983.; pp. 109-114.

[7] Goodman, L.S., "Protocol # L-163: Tilt-Table Simulation of Push-Pull Effect and Tilt-Table Study, Squat-Stand Test and Risk Factor Investigation", Defence and Civil Institute of Environmental Medicine, Downsview, Ontario, January 1997.

[8] Banks, R.D., J.D. Grissett, A.J. Mateczun., "The Effects of Varying Times at $-G_z$ on Subsequent $+G_z$ Physiological Tolerance (Push-Pull Effect)", *Aviat Space Environ Med* 1995; 66:723-727.

[9] Banks, R.D., J.D. Grissett, G.T. Turnspeed, et al., "The 'push-pull' effect", *Aviat Space Environ Med* 1994; 65:699-704.

[10] "Development of an Expert System for Automatic Generation and Adaptive Optimization of Push-Pull Countermeasures for Typical and Atypical Pilots, Phase XI", *Fi-*

nal Report prepared for the Defence and Civil Institute of Environmental Medicine, Downsview, Ontario, 1999.

List of Appendices

Appendix A: ESIstat R Source code

Appendix B: ESIstat R Documentation

Appendix C: ESIstat Tutorials

Appendix D: Experimental Protocol #L-239

```
 #Includes functions
firsttry, refirsttry, last, P239barplot, P239meanbarplot
#P239plot, P239boxplot, stackandbox, calcslope, firstquart,
#ecgp2rr, ecgp2hr, syadia2map, sysdia2pp, sysdia2si #P239pairwise,
P239.t.test,P239subset,HonestlySigDiff, groupdata

firsttry<-function(TSTATobject,Channel, GZtimes,win,
          windowlength,startnum=1,endnum=207,applyfunction)
{




 if (!any(win==c("pre","post","hdt","pre&hdt")))
 {
  cat("Window must be Either pre or post \n");
 }
 else
 {
  if (win  == "pre")
  {
   useGZtimes<-cbind(GZtimes[,1]-windowlength,GZtimes[,1]);
   result<-Dfapplyfun("calcDFC",useGZtimes,TSTATobject,
    "Whole",Channel,startnum,endnum, applyfunction);
  }

  if (win == "post")
  {
   useGZtimes<-cbind(GZtimes[,3],GZtimes[,3]+windowlength);
   result<-Dfapplyfun("calcDFC",useGZtimes,TSTATobject,
    "Whole",Channel,startnum,endnum, applyfunction);
  }

  if (win == "hdt")
  {
   useGZtimes<-cbind(GZtimes[,2],GZtimes[,3]);
   result<-Dfapplyfun("calcDFC",useGZtimes,TSTATobject,
    "Whole",Channel,startnum,endnum, applyfunction);
  }

  if (win == "pre&hdt")
  {

   useGZtimes<-cbind(GZtimes[,1]-windowlength,GZtimes[,2]+windowlength);
   result<-Dfapplyfun("calcDFC",useGZtimes,TSTATobject,
    "Whole",Channel,startnum,endnum, applyfunction);
  }


 }
return(result);
}


refirsttry<-function(TSTATobject,Channel, GZtimes,win,
    windowlength,startnum=1,endnum=207,applyfunction)
{
```

```
if (win != "pre" && win != "post" && win != "hdt" && win !="pre&hdt")
{
 cat("Window must be Either pre or post \n");
}
else
{
 if (win  == "pre")
 {
  useGZtimes<-cbind(GZtimes[,1]-windowlength,GZtimes[,1]);
  result<-Dfapplyfun("recalcDFC",useGZtimes,TSTATobject,
   "Whole",Channel,startnum,endnum,applyfunction);
 }

 if (win == "post")
 {
  useGZtimes<-cbind(GZtimes[,3],GZtimes[,3]+windowlength);
  result<-Dfapplyfun("recalcDFC",useGZtimes,TSTATobject,
   "Whole",Channel,startnum,endnum,applyfunction);
 }

 if (win == "hdt")
 {
  useGZtimes<-cbind(GZtimes[,2],GZtimes[,3]);
  result<-Dfapplyfun("recalcDFC",useGZtimes,TSTATobject,
   "Whole",Channel,startnum,endnum,applyfunction);
 }
```

```
  if (win == "pre&hdt")
  {

   useGZtimes<-cbind(GZtimes[,1]-windowlength,GZtimes[,2]+windowlength);
   result<-Dfapplyfun("recalcDFC",useGZtimes,TSTATobject,
    "Whole",Channel,startnum,endnum,applyfunction);
  }


 }
return(result);
}

last<-function(x)
{

return(x[length(x)]);

}

P239barplot<-function(FileName,DF,subjectcol)
{
#This function produces series of barplots,
#For each subject in "subjectcol", four plots are produced:
#  Delta SYS BP by runtype
#  Delta DIA BP by runtype
#  Delta RR by runtype
#  Delta SI by runtype
#Currently, the code expects the delta values to be in
#specific columns (3,6,9,12) and the Runtypes to be in
#column 14. This could be easily generalized
#The bars are ordered in increasing value
varnames<-names(DF);
DF<-na.omit(DF);

temppar<-par(no.readonly=T);
par(ask=T);
subnames<-as.vector(unique(DF[,subjectcol]))

##FileName<-"FirstTry";
TableTitle<-scan(file="",what=character(0),nmax=1);
##numcols<-as.numeric(scan(file="",what=character(0),nmax=1));
numcols<-2;

TexStartTableFig(FileName,4,TableTitle);

for(k in 1:length(subnames))
{
 for (i in c(3,6,9,12))
 {
  data<-DF[DF[,14]==subnames[k],i];
  barnames<-as.character(DF[DF[,14]==subnames[k],13]);
  temp<-order(data);
  barplot(data[temp],names.arg=barnames[temp],col="blue");
  title(main=paste("Barplot of ",varnames[i]," for Subject ",
  subnames[k],sep=""),ylab=varnames[i])
  TexTableFig(FileName,"");
 }
}

TexEndTableFig(FileName);
par(temppar);
}

P239meanbarplot<-function(FileName,DF,ordervect,SEMs,ascending=F)
{
#This function produces series of barplots,
#Similar to P239barplot except the subject pool means are plotted.
#For each subject pool, four plots are produced:
#  Delta SYS BP by runtype
#  Delta DIA BP by runtype
#  Delta RR by runtype
#  Delta SI by runtype
#Currently, the code expects the delta values to be in
#specific columns (4,7,10,13) and the Runtypes to be in
#column 14.  This could be easily generalized
#The bars are ordered in increasing value
#An additional input, ordervect is expected, which assigns
#a number to each column for ordering purposes
#the mean values data frame was generated via (for example)
#firstDFpoolmeans<-aggregate.data.frame(firstDF[,-c(13,14),
#   by=list(firstDF$ExpRuns),FUN=mean)
temppar<-par(no.readonly=T);
```

```
varnames<-names(DF);
#which(is.na())
#DF<-na.omit(DF);

par(ask=T, las=3);

##FileName<-"FirstTry";
cat("Enter Table Title ");
TableTitle<-scan(file="",what=character(0),nmax=1);
##numcols<-as.numeric(scan(file="",what=character(0),nmax=1));
numcols<-2;

TexStartTableFig(FileName,2,TableTitle);

for (i in c(4,7,10,13))
# for(i in c(13))
 {
  data<-DF[ordervect,i];
  dataSEM<-SEMs[ordervect,i];
  barnames<-as.character(DF[ordervect,1]);

  omitindices<-which(is.na(data));
  #return(cbind(barnames,data));
  if (length(omitindices)!=0)
  {
  data<-data[-omitindices];
  barnames<-barnames[-omitindices];
  dataSEM<-dataSEM[-omitindices];
  }
if(any(data<0)) yrange<-c(min(data)-max(dataSEM),max(data)+max(dataSEM))
else  yrange<-c(0,max(data)+max(dataSEM));
##  yrange<-c(0,60);
  if(ascending){
  temp<-order(data);
  data<-data[temp];
  dataSEM<-dataSEM[temp];
  barnames<-barnames[temp];
  }
  barpts<-barplot(data,names.arg=barnames,col="blue",ylim=yrange);
  title(main=paste("Barplot of ",varnames[i]," for Subject
    Pool Means ",sep=""),ylab=paste(varnames[i],"+/- SEM",sep=" "))

  barwidth<-(barpts[2]-barpts[1])*0.8/2;

  for(j in 1:length(barpts))  #add SEM bars
  {
  if(data[j]>0)
   {
   lines(c(barpts[j],barpts[j]),c(data[j],data[j]+dataSEM[j]),col="red");
   lines(c(barpts[j]-barwidth,barpts[j]+barwidth),c(data[j]+
    dataSEM[j],data[j]+dataSEM[j]),col="red");
   }
  else
   {
   lines(c(barpts[j],barpts[j]),c(data[j],data[j]-dataSEM[j]),col="red");
   lines(c(barpts[j]-barwidth,barpts[j]+barwidth),c(data[j]-
    dataSEM[j],data[j]-dataSEM[j]),col="red");
   }
  }
  TexTableFig(FileName,"");
##return(dataSEM);
##return(barpts);
 }

TexEndTableFig(FileName);
par(temppar);
}



P239plot<-function(DF,x1,x2,xfacts,ylabel,xlabel)
{
##this function
temppar<-par(no.readonly=T);
FileName<-"FirstTry";
TableTitle<-scan(file="",what=character(0),nmax=1);
##numcols<-as.numeric(scan(file="",what=character(0),nmax=1));
numcols<-4;
TexStartTableFig(FileName,4,TableTitle);
xfactlevels<-as.character(unique(DF[,xfacts]));
ymax<-max(c(as.vector(DF[,x1]),as.vector(DF[,x2])),na.rm=TRUE);
yrange<-c(0,ymax);
##return(yrange);
par(xaxp=c(0,1,1),xaxt="n");
```

```
for(i in 1:length(xfactlevels))
 {
 y1<-DF[DF[,xfacts]==xfactlevels[i],x1];
 y2<-DF[DF[,xfacts]==xfactlevels[i],x2];
 plot.new()
 par(ask=F)
 for (j in 1:length(y1))
  {
  par(new=T);
  plot(c(0,1),c(y1[j],y2[j]),type="o",ylim=yrange,xlab="",ylab="");
  }
 axis(1,0:1,c(names(DF)[x1],names(DF)[x2]),xaxt="s",las=3);
 title(main=xfactlevels[i],ylab=ylabel,xlab=xlabel);
 TexTableFig(FileName,"");
 par(ask=T)
}
TexEndTableFig(FileName);
par(temppar);
}

ecgp2rr<-function(ECGPvector)
{
#As the name suggests, this function takes a vector of ECGP (ECG peaks)
#data obtained by TSTAT and returns a vector of RR intervals with timestamps
 if (length(ECGPvector)==0) return(array(c(NA,NA),dim=c(1,2)));
 rr<-c()
 ##simple function to compute the rr interval, given the ECG peak data
 numrr<-length(ECGPvector[,1])-1;
 for (i in 1:numrr)
 {
  rr<-c(rr,ECGPvector[i+1,1]-ECGPvector[i,1])
 }
 return(cbind(ECGPvector[1:numrr,1],rr))
}

ecgp2hr<-function(ECGPvector)
{
#Similar to ecgp2rr, this function takes a vector of ECGP (ECG peaks)
#data obtained by TSTAT and returns a vector of Heart Rates with timestamps

 if (length(ECGPvector)==0) return(array(c(NA,NA),dim=c(1,2)));
 hr<-c()
 numhr<-length(ECGPvector[,1])-1;
 ##simple function to compute the heart rate, given the ECG peak data
 for (i in 1:numhr)
 {
  hr<-c(hr,1/(ECGPvector[i+1,1]-ECGPvector[i,1]))  ##Heart rate in BPS
 }
 return(cbind(ECGPvector[1:numhr,1],hr))
}

sysdia2pp<-function(SYS,DIA)
{
#As the name suggests, this function takes a vector of SYS (Systolic
#Blood Pressure) and DIA (Diastolic Blood Pressure) data obtained by TSTAT
#and returns a vector of Pulse Pressures with timestamps
```

```
 if (length(SYS)==0 || length(DIA)==0) return(array(c(NA,NA),dim=c(1,2)));
 pp<-c();
 numDIA<-length(SYS[,1])-1;
 ##simple function to compute the pulse pressure, given the SYS and DIA data
 if (SYS[1,1] < DIA[1,1]) {startpt<-2; shift<-1}
 else {startpt<-1; shift<-0;}

 for (i in startpt:numDIA)
 {
  pp<-c(pp,(SYS[i,2]-DIA[i-shift,2]))  ##Pulse Pressure
 }
 return(cbind(DIA[1:numDIA,1],pp))
}

sysdia2map<-function(SYS,DIA)
{
#As the name suggests, this function takes a vector of SYS (Systolic
#Blood Pressure) and DIA (Diastolic Blood Pressure) data obtained by TSTAT
#and returns a vector of mean arterial Pressures with timestamps

 if (length(SYS)==0 || length(DIA)==0) return(array(c(NA,NA),dim=c(1,2)));
 map<-c();
 numDIA<-length(SYS[,1])-1;
 ##simple function to compute the pulse pressure, given the SYS and DIA data
 if (SYS[1,1] < DIA[1,1]) {startpt<-2; shift<-1}
 else {startpt<-1; shift<-0;}
 for (i in startpt:numDIA)
 {
  map<-c(map,(SYS[i,2]+2*DIA[i-shift,2])/3)  ##MAP
 }
 return(cbind(DIA[1:numDIA,1],map))
}

sysdia2si<-function(SYS,DIA)
{
#As the name suggests, this function takes a vector of SYS (Systolic
#Blood Pressure) and DIA (Diastolic Blood Pressure) data obtained by TSTAT
#and returns a vector of Stiffness Indexes with timestamps

 if (length(SYS)==0 || length(DIA)==0) return(array(c(NA,NA),dim=c(1,2)));
 si<-c();
 numDIA<-length(SYS[,1])-1;
 ##simple function to compute the Stiffness Index,
 ##given the SYS and DIA data
 if (SYS[1,1] < DIA[1,1]) {startpt<-2; shift<-1}
 else {startpt<-1; shift<-0;}

 for (i in startpt:numDIA)
 {
  si<-c(si,(SYS[i,2]-DIA[i-shift,2])/(SYS[i,1]-
    DIA[i-shift,1])); ##Stiffness Index
 }
 return(cbind(DIA[1:numDIA,1],si))
}
```

```
P239boxplot<-function(DF,Dep,Indep,logvect=rep(T,26))
{
#This function is used to prepare boxplots of a DataFrame (DF)
#The Dep input indicates the Dependant variable to be plotted on the y axis
#The Indep input indicates the Independent variable to
#be plotted along the x axis
#The plot parameters are read beforehand and reset afterwards

temppar<-par(no.readonly=T);
boxcolors<-c(rep("lightblue",3),rep("pink",4),
    rep("yellow",3),rep("cyan",8),rep("tan",6),
    rep("wheat",2));
Dep<-which(names(DF)==Dep);
Indep<-which(names(DF)==Indep);

boxplot(DF[,Dep]~DF[,Indep],colors=boxcolors[logvect]);
par(temppar);
}

firstquart<-function(x)
{#A very simple funtion to return the first quartile of a vector of data
return(quantile(x,1/4));}

calcslope<-function(y,x)
{#A very simple function to calulate the slope of a set of data points (x,y)
return(lm(y~x));}
```

```
stackandbox<-function(DataFrame,subnames="")
{
#This function is used to rearrange a data frame of dynamic points into a
#form suitable for boxplotting and/or analyses of variance.
#The first two rows are ignored since they are assumed to be
#timestamps and GZ data
#Each column is stacked below the previous to form a long data frame to
#which a vector of Factors can be cbinded
#The rownames are repeated as the rownames of the resulting data frame
#At the end the result is boxplotted against the rownames,
#and the stacked data frame returned

temp<-rownames(DataFrame);
DataFrame<-DataFrame[3:(dim(DataFrame)[1]),]  #Discard first two rows
rownames(DataFrame)<-temp[3:length(temp)];    #Set names
newtemp<-c();
newtempnames<-c();
newtempsubjects<-rep(as.character(subnames),dim(DataFrame)[2]);
newtempfacts<-c();

for(i in 1:(dim(DataFrame)[2]))
{
 newtemp<-c(newtemp,as.vector(DataFrame[,i]));
 newtempfacts<-c(newtempfacts,rep(i,dim(DataFrame)[1]));
 #factors are simply the column number (ie time)
}
for( j in 1:(dim(DataFrame)[2]))
 {
 newtempnames<-c(newtempnames,rownames(DataFrame));
 }

newtemp<-as.data.frame(newtemp);
newtempnames<-as.data.frame(newtempnames);
newtemp<-cbind(newtemp,newtempnames); ##newtempfacts);
rm(newtempnames);
##plotindices<-(newtempsubjects!="S4" & newtempsubjects!="S7")
boxplot(newtemp[,1]~newtemp[,2])
return(newtemp);
}

P239pairwise<-function(DF,bycol,subsetval,method="bonf")
{
#This function is used to perform multiple pairwise
#comparisons on a subset of a data frame
#The data frame is assumed to be in the form output by stackandbox
#The subset column is specified in "bycol" and the subset
#values in "subsetval" (as a vector)
#the p.adjust.method can be passed via "method" but is
#by default "bonferroni"
temp<-P239subset(DF,bycol,subsetval);
return(pairwise.t.test(temp[,1],as.character(temp[,2]),
    p.adjust.method=method,pool.sd=F))
}
```

```
P239.t.test<-function(DF,datacol,bycol,subsetval)
{
#This function performs a t.test on a data frame
#To select the desired subset, pass the vector of factors in "subsetval"
#
temp1<-P239subset(DF,bycol,subsetval[1]);
temp1<-temp1[,datacol];
##return(temp1);
temp2<-P239subset(DF,bycol,subsetval[2]);
temp2<-temp2[,datacol];
return(t.test(temp1,temp2,alternative="two.sided"))
}

P239subset<-function(DF,subsetcol,subsetval)
{
subsetDF<-data.frame()
for( i in 1:length(subsetval))
 {
##return(DF[DF[,subsetcol]==subsetval[i],]);
subsetDF<-rbind(subsetDF,DF[DF[,subsetcol]==subsetval[i],]);
 }
return(subsetDF);
}

HonestlySigDiff<-function(DF,bycol)
{
nmeans<-length(unique(as.character(DF[,bycol])))
#nmeans is the number of means being compared
```

```
nmeans<-5;  #
df<-40;     #
MSerror<-9.67;   #temporary
n<-10;      #

##if number of observations for each mean are not the same,
##calculate harmonic mean

qvalue<-qtukey(0.95,nmeans,df,1)
HSD<-qvalue*sqrt(MSerror/n)  #n is num of observations making up each mean
                 #MSerror is the mean square error from within groups
return(HSD);
}

groupdata<-function(DF,bycol,Group1,Group2,Groupnames)
{
result<-as.data.frame(DF[1:2,]);
resultfact<-c("GZ","TimeStamp");
temp<-c()

if(bycol=="rownames")bycoldata<-rownames(DF)
else bycoldata<-as.vector(DF[,bycol]);
#return(bycoldata);
for(i in 1:(dim(DF)[1]))
{
if(any(bycoldata[i]==Group1)) {
    result<-rbind(result,DF[i,]);
    resultfact<-c(resultfact,Groupnames[1]);
}
if(any(bycoldata[i]==Group2)) {
    result<-rbind(result,DF[i,]);
    resultfact<-c(resultfact,Groupnames[2])
}
#return(as.data.frame(cbind(DF[i,],Groupnames[2])));
}
rownames(result)<-resultfact;
return(result);
#return(cbind(result,resultfact));
}


##DFapplyfun, callTSTAT, Vdecimate, Vintegratefull,addfacts,addnumeric,
##Gzpoints
##VerifyDFC,dynamicpts,dynplotdelta,dynplot,TSTATgetdata


dynamicpts<-function(TSTATobject,Timepts,ChName,plot=TRUE)
{
##TSTATobject and ChName are specified as strings, Timepts is a numeric
##vector or matrix.The first column of Timepts must contain the point to
##syncronize to (usually start of Tilt) for each insertion.  Any other
##columns are ignored.
##ChName is the channel name for the datatrac file
##(or custom names "rr","hr", "pp")
##This function synchronizes the data from each insertion,
##according to the time in Timepts, and interpolates
##the values each second
##The routine accesses the "Whole" profile of TSTAT, and uses the start
```

```
##times and Timepts to determine which insertion has the least amount of
##data before the tilt. Similarly, the least amount of data after tilt is
##determined.  All insertions are processed using these time
##limits. This does discard data from some insertions, but avoids the
##problems associated with NA's later on.

TSTATinfo<-callTSTAT(TSTATobject,"get");
insertions<-names(TSTATinfo$data)

 maxendtime<-0;
 minendtime<-1000;
 maxGZstart<-max(Timepts[,1]);
 minGZstart<-min(Timepts[,1]);
 result<-c()

 ##find range for data
 for (i in 1:length(insertions))
 {
 curendpt<-eval(parse(text=paste("TSTATinfo$data$",
    insertions[i],"$Profiles$data$Whole$End",sep="")))
 curstartpt<-eval(parse(text=paste("TSTATinfo$data$",
    insertions[i],"$Profiles$data$Whole$Start",sep="")))
# curendpt<-curendpt-curstartpt+maxGZstart-Timepts[i,1];
 curendpt<-curendpt-curstartpt+minGZstart-Timepts[i,1];
# if (curendpt>maxendtime){ maxendtime<-curendpt; ins<-i;}
 if (curendpt<minendtime){ minendtime<-curendpt; ins<-i;}
 }
# timestamps<-(1:round(maxendtime));  #time pts every second
```

```
 timestamps<-(1:round(minendtime));  #time pts every second

 result<-rbind(result,timestamps);

if(plot) {par(ask=T)};

for (i in 1:length(insertions))
##for (i in 104:104)
 {##get data and save to array
 callTSTAT(TSTATobject,"select",insertions[i]);
 data<-TSTATgetdata(TSTATobject,"Whole",ChName)
# data[,1]<-data[,1]+maxGZstart-Timepts[i,1];
 data[,1]<-data[,1]+minGZstart-Timepts[i,1];

if(plot) {
plot(data,type="b",col="black",xlim=c(0,minendtime),
    main=insertions[i],ylab="Channel data", xlab="Time(s)");
if(menu(c("Exclude Points"),title="Select one to exclude points")==1)
 {excludepts<-identify(data[,1],data[,2],plot=T);

 if (length(excludepts) !=0) data<-data[-excludepts,];
 }
}

fc<-approxfun(data[,1],data[,2],method="linear")
data1<-fc(timestamps);
##fc<-splinefun(data[,1],data[,2],method="natural")
##data1<-fc(timestamps)
```

```
 GZdata<-TSTATgetdata(TSTATobject,"Whole","GZ")
# GZdata[,1]<-GZdata[,1]+maxGZstart-Timepts[i,1];
 GZdata[,1]<-GZdata[,1]+minGZstart-Timepts[i,1];

#There must be a better way than this but it works for now!
 if (i==2)
 {
 fc<-approxfun(GZdata[,1],GZdata[,2],method="linear")
 GZreturndata<-fc(timestamps);
 }
 yrange<-range(data[,2])

#changed maxendtimes to minendtimes
if(plot) {
plot(GZdata,col="lightgrey",xlim=c(0,minendtime),
    ylim=c(0,150),main=insertions[i],xlab="",ylab="",yaxt="n")
 par(new=T,ask=F);
 plot(data,xlim=c(0,minendtime),col="black",ylim=yrange,xlab="",ylab="");
 par(new=T);

plot(cbind(timestamps,data1),type="b",col="green",
    ylim=yrange,xlim=c(0,minendtime),ylab="Channel data",
    xlab="Time(s)");

 par(ask=T)
}
```

```
 result<-rbind(result,data1);
 }
 if (exists("GZreturndata")){ result<-rbind(GZreturndata,result);}

 return(as.data.frame(result))
}



dynplotdelta<-function(dyndata,restlength,showcases="all",
    maintitle="", xl="",yl="")
{
#This routine is essentially the same as dynplot, except that
#it first calculates the delta values for the second by second data
#The reference rest level is calculated from the first
#"restlength data points for each insertion
##plotcolors<-c("blue","red","green","black","yellow","darkgrey",
##"orange","purple","lightblue","grey","lightgrey","brown","cyan",
##"tan","pink","darkblue","darkred","darkorange","olivedrab","gold",
##"steelblue","yellowgreen","peachpuff4","chartreuse","limegreen",
##"navy",rep("green",24))
plotletters<-c("A","B","C","D","E","F","G","H","I","J","K","L","M",
    "N","O","P","Q","R","S","T","U","V","W","X","Y","Z")
plotcolors<-c("red","black","green","blue")
plotsymbols<-c("circle","triangle","plus","X","diamond","upsidedowntri",
    "squareX","asterisk","diamondX","circleplus","twotriangles",
    "squareplus","circleX","squaretriangle","solidsquare",
    "solidcircle","solidtriangle","soliddiamond")
```

```
refchannel<-dyndata[1,];
numdata<-dim(dyndata)[1];
lengthdata<-dim(dyndata)[2];
deltadata<-dyndata[1:2,];
restmean<-c();



for( i in 3:(dim(dyndata)[1]))
 {
 tempdelta<-c()
 restmean<-mean(dyndata[i,1:restlength]);
 for (j in 1:(dim(dyndata)[2]))
  {
  tempdelta<-c(tempdelta,dyndata[i,j]-restmean);
  }
 deltadata<-rbind(deltadata,tempdelta)
 }

##deltadata<-rbind(dyndata[1:2,],deltadata);

rownames(deltadata)<-rownames(dyndata);
#
dyndata<-deltadata;
rm(deltadata);
#return(dyndata);
meanvals<-c();
stdvals<-c()
```

```
casenames<-unique(rownames(dyndata)[3:length(rownames(dyndata))]);
##casenames<-unique(rownames(dyndata[3:(dim(dyndata)[1]),]));
#casenames<-unique(rownames(dyndata));
#casenames<-
#return(casenames);
for (i in 1:length(casenames))
{
 casedata<-dyndata[rownames(dyndata)==casenames[i],];
 casemean<-c();
 casestd<-c();
 for (j in 1:(dim(dyndata)[2]))
 {
 casemean[j]<-mean(casedata[,j],na.rm=T);
 casestd[j]<-sd(casedata[,j],na.rm=T);
 casestd[j]<-casestd[j]/sqrt(length(casedata[,j])); ##convert to SEM
##return(c(length(casedata[,j]),sqrt(length(casedata[,j]))));
 }

meanvals<-rbind(meanvals,casemean)
stdvals<-rbind(stdvals,casestd)

}

rownames(meanvals)<-casenames[1:length(casenames)];
rownames(stdvals)<-casenames[1:length(casenames)];
#return(stdvals);

par(ask=T,new=F)
```

```
plot(as.numeric(refchannel),type="l",col="lightgrey",
    xlab="",ylab="",yaxt="n");
#return()
docases<-c()
if (showcases=="all") docases<-(1:length(casenames))
else
 {
  for(i in 1:length(showcases))
  {
   docases<-c(docases,which(rownames(meanvals)==showcases[i]));
  }
 }
cat(paste("Cases are:",rownames(meanvals)[docases],
    " (",plotcolors[1:length(docases)],")\n"))
##return(docases);
for (i in docases)
 {
 par(new=T);

plotcol<-plotcolors[which(docases==i)]
plot(meanvals[i,],pch=which(docases==i),col=plotcol,
    main=maintitle,xlab=xl,ylab=yl,ylim=c(-30,40));
 for (j in 1:length(meanvals[i,]))
 {

lines(c(j,j),c(meanvals[i,j]+stdvals[i,j],meanvals[i,j]-
    stdvals[i,j]),col=plotcol,lty=which(docases==i))
lines(c(j-0.2,j+0.2),c(meanvals[i,j]+stdvals[i,j],
    meanvals[i,j]+stdvals[i,j]),col=plotcol,lty=which(docases==i))

lines(c(j-0.2,j+0.2),c(meanvals[i,j]-stdvals[i,j],
    meanvals[i,j]-stdvals[i,j]),col=plotcol,lty=which(docases==i))
 }

}

title(main=paste(rownames(meanvals)[docases],collapse=
    " and "),ylab="Pool Mean +/- SEM",xlab="Time",
    sub=paste(rownames(meanvals)[docases]," (",
    plotcolors[1:length(docases)]," ",
    plotsymbols[1:length(docases)],")   ",collapse=""))
return(dyndata);
}




dynplot<-function(dyndata,showcases="all",maintitle="",
    xl="",yl="", ylim=c(20,200))
{
#dyndata should be a numeric matrix with each row containing
#time series data at the same sample times (generated by dynamicpts())
#The routine skips the first two rows since they are assumed to contain
#reference channel data (GZ) and timestamps, as output from dynamicpts.
#The routine uses the rownames of dyndata to group data and
#calculate pooled data this function calculates the group pool mean
```

```
#and standard deviation and plots the results for the cases in showcases


##plotcolors<-c("blue","red","green","black","yellow","darkgrey",
##"orange","purple","lightblue","grey","lightgrey","brown","cyan",
##"tan","pink","darkblue","darkred","darkorange","olivedrab","gold",
##"steelblue","yellowgreen","peachpuff4","chartreuse","limegreen","navy")
plotletters<-c("A","B","C","D","E","F","G","H","I","J","K","L",
    "M","N","O","P","Q","R","S","T","U","V","W","X","Y","Z")

#Colors used for plotting.  If more than four groups are to be plotted,
#more colors must be added here
plotcolors<-c("red","black","green","blue")

#Names of symbols used for legend
plotsymbols<-c("circle","triangle","plus","X","diamond","upsidedowntri",
    "squareX","asterisk","diamondX","circleplus","twotriangles",
    "squareplus","circleX","squaretriangle","solidsquare",
    "solidcircle","solidtriangle","soliddiamond");

dyndata <- as.matrix(dyndata)
refchannel<-dyndata[1,];
numdata<-dim(dyndata)[1];
lengthdata<-dim(dyndata)[2];
##dyndata<-dyndata[3:(dim(dyndata)[1]),];
##dyndata<-cbind(dyndata,byvector);

meanvals<-c();
```

```
stdvals<-c()
casenames<-unique(rownames(dyndata));
for (i in 3:length(casenames))
{
 casedata<-dyndata[rownames(dyndata)==casenames[i],];
 casemean<-c();
 casestd<-c();
 for (j in 1:(dim(dyndata)[2]))
 {
 casemean[j]<-mean(casedata[,j],na.rm=T);
 casestd[j]<-sd(casedata[,j],na.rm=T);
 casestd[j]<-casestd[j]/sqrt(length(casedata[,j])); ##convert to SEM
 }

meanvals<-rbind(meanvals,casemean)
stdvals<-rbind(stdvals,casestd)

}

rownames(meanvals)<-casenames[3:length(casenames)];
rownames(stdvals)<-casenames[3:length(casenames)];

#This mess was needed previously to omit NA's before dynamicpts was modified
##meanvals<-aggregate(dyndata[3:numdata,],by=list(byvector),
##  FUN=mean,na.rm=T);
##stdvals<-aggregate(dyndata[3:numdata,],by=list(byvector),FUN=sd,na.rm=T);
##casenames<-unique(byvector);##as.character(meanvals[1,]);
##meanvals<-array(meanvals[,2:(dim(meanvals)[2])]);
```

```
##stdvals<-array(stdvals[,2:(dim(stdvals)[2])]);
#meanvals<-t(meanvals);
#return(meanvals);
#meanvals<-na.omit(meanvals);
#meanvals<-t(meanvals);
#stdvals<-t(stdvals);
#stdvals<-na.omit(stdvals);
#stdvals<-t(stdvals);
#return(stdvals);
#omitindices<-as.vector(attributes(stdvals)[3]$na.action);
#refchannel<-refchannel[-omitindices];
#refchannel<-na.omit(refchannel);
##return(refchannel);
#timevect<-1:dim(dyndata)[2];


par(ask=T,new=F)
plot(refchannel,col="lightgrey",type="line",xlab="",ylab="",yaxt="n");
docases<-c()
if (showcases=="all") docases<-(1:length(casenames))
else
 {
  for(i in 1:length(showcases))
  {
   docases<-c(docases,which(rownames(meanvals)==showcases[i]));
  }
 }
cat(paste("Cases are:",rownames(meanvals)[docases]," (",
    plotcolors[1:length(docases)],")\n"))
##return(docases);
for (i in docases)
 {
 par(new=T);
 plotcol<-plotcolors[which(docases==i)];

 plot(meanvals[i,],pch=which(docases==i),col=plotcol,
    main=maintitle,xlab=xl,ylab=yl,ylim=ylim);
 for (j in 1:length(meanvals[i,]))
 {

lines(c(j,j),c(meanvals[i,j]+stdvals[i,j],meanvals[i,j]-
    stdvals[i,j]),col=plotcol,lty=which(docases==i))

lines(c(j-0.2,j+0.2),c(meanvals[i,j]+stdvals[i,j],
    meanvals[i,j]+stdvals[i,j]),col=plotcol,lty=which(docases==i))

lines(c(j-0.2,j+0.2),c(meanvals[i,j]-stdvals[i,j],
    meanvals[i,j]-stdvals[i,j]),col=plotcol,lty=which(docases==i))
 }

 }

title(main=paste(rownames(meanvals)[docases],collapse=" and "),
    ylab="Pool Mean +/- SEM",xlab="Time",sub=paste(rownames(meanvals)[docases],
    " (",plotcolors[1:length(docases)]," ",plotsymbols[1:length(docases)],
    ")   ",collapse=""))
```

```
}

TSTATgetdata<-function(TSTATname,ProfName,ChName)
{
##This is a routine to get data from a datatrac file, using TSTAT
##note: the proper insertion should be set before calling
##this function using the TSTAT function $set()
##TSTATname is a character string, as is ProfName which
##is the name of the profile in TSTAT
##ChName is the desired Channel.  It must be either the proper TAG
##for the Datatrac file or a custom one which is handled here
##  (currently "rr" for RR interval, "hr" for heart rate,
##  "pp" for pulse pressure
##  note, by convention, UPPER CASE letters are used for proper datatrac
##  tags, while lower case are used for channels that must be calculated

######## Get Data for channel ###############
if(ChName =="rr" || ChName =="hr" ||ChName=="pp" || ChName =="si" )
    ###Datatrac files don't have RR or HR data
{
## data<-callTSTAT(TSTATname,"getd",ProfName,ChName)


 if(ChName =="hr")
 {
 ecgpdata<-callTSTAT(TSTATname,"getd",ProfName,"ECGP");
 ecgdata<-callTSTAT(TSTATname,"getd",ProfName,"ECG")
 data<-ecgp2hr(ecgpdata);
 }

 if(ChName =="rr")
 {
 ecgpdata<-callTSTAT(TSTATname,"getd",ProfName,"ECGP");
 ecgdata<-callTSTAT(TSTATname,"getd",ProfName,"ECG")
 data<-ecgp2rr(ecgpdata);
 }

 if(ChName=="pp")  ###Datatrac files don't have Pulse Pressure Data
 {
 SYSdata<-callTSTAT(TSTATname,"getd",ProfName,"SYS");
 DIAdata<-callTSTAT(TSTATname,"getd",ProfName,"DIA");
 data<-sysdia2pp(SYSdata,DIAdata);
 rm(SYSdata,DIAdata);
 gc()
 }

 if(ChName=="si")  ###Datatrac files don't have Pulse Pressure Data
 {
 SYSdata<-callTSTAT(TSTATname,"getd",ProfName,"SYS");
 DIAdata<-callTSTAT(TSTATname,"getd",ProfName,"DIA");
 data<-sysdia2si(SYSdata,DIAdata);
 rm(SYSdata,DIAdata);
 gc()
 }

}
```

```
else                    ###All other channels are in datatrac file
{
 data<-callTSTAT(TSTATname,"getd",ProfName,ChName);
}
return(data)

}

Dfapplyfun<-function(FunName,ExtraVect,TSTATname,ProfName,
    ChName,startnum=0,endnum=0,...)
{

##This is a generic sort of function to loop through the
##insertions in a TSTAT object.
##It is useful for setting
##The retunred vector of data contains the same number of rows (in
##the same order) as the TSTAT object
##The strings ProfName and ChName define the data to extract using TSTAT
##The string FunName is the function that will be applied to the data
##ExtraVect (which can actually be an array )is a general use argument
##which is used to pass various type of data to different functions
##startnum and endnum refer to the TSTAT insertion numbers
##(or data frame rows) to process
##extra arguments can be passed through .. to the functions.

par(ask=T,mfrow=c(2,1));
TSTATinfo<-callTSTAT(TSTATname,"get");
insertions<-names(TSTATinfo$data);
```

```
if (startnum==0 || endnum==0){startnum<-1;endnum<-length(insertions);}
temp<-c();
result<-c();
decfactor<-10;
##for (i in 1:length(insertions)) ##rownames(DFName)))
##for (i in c(5,8,9,14,15,31,34,35,40,41,57,60,61,66,67,83,86,
##  87,92,93,109,112,113,118,119,135,138,139,144,145,161,
##  164,165,170,171,186,189,190,195,196))
##for(i in c(2,28,54,80,106,132,158,184))
for(i in startnum:endnum)
{


par(new=F,ask=T); ##should be ask=T
insertion<-insertions[i];
callTSTAT(TSTATname,"select",insertion);

##data<-TSTATgetdata(TSTATname,ProfName,ChName)

data<-TSTATgetdata(TSTATname,ProfName,ChName);
######## Get Data for channel ###############
##This could be easily implemented using TSTATgetdata()
##but it was developed here first
#if(ChName =="rr" || ChName =="hr" || ChName =="pp")
###Datatrac files don't have RR, HR , or pulse pressuredata
#{
#
```

```
# if(ChName =="hr")
# {
# ecgpdata<-callTSTAT(TSTATname,"getd",ProfName,"ECGP");
# ecgdata<-callTSTAT(TSTATname,"getd",ProfName,"ECG");
# data<-ecgp2hr(ecgpdata);
# }
#
# if(ChName =="rr")
# {
# ecgpdata<-callTSTAT(TSTATname,"getd",ProfName,"ECGP");
# ecgdata<-callTSTAT(TSTATname,"getd",ProfName,"ECG");
# data<-ecgp2rr(ecgpdata);
# }
#
# if(ChName=="pp")
# {
# SYSdata<-callTSTAT(TSTATname,"getd",ProfName,"SYS");
# DIAdata<-callTSTAT(TSTATname,"getd",ProfName,"DIA");
# data<-sysdia2pp(SYSdata,DIAdata);
# rm(SYSdata,DIAdata);
# gc();
# }
#
#
#}
#
#else   ###Special cases are handled in if case All other
###channels are in datatrac file and are handled here
```

```
#{
# data<-callTSTAT(TSTATname,"getd",ProfName,ChName);
#}

######### Decimate Data ########################
## if (ChName != "SYS" && ChName != "DIA" && ChName !="ECG" &&
##  ChName !="ECGP" && ChName != "rr" && ChName != "hr" &&
##  ChName !="pp")
if (!any(ChName==c("SYS","DIA","ECG","ECGP","rr","hr","pp")))

 ### Don't decimate variable rate data
 {
   datavalues<-Vdecimate(data[,2],decfactor);
   datatimes<-Vdecimate(data[,1],decfactor);
 }
 else
 {
  datavalues<-data[,2];
  datatimes<-data[,1];
 }
 ##return(datatimes);
 rm(data);
 if (!is.na(datavalues)) {plotylim<-c(0,max(datavalues));}
 else {plotylim<-c(0,100);}


######### Always get GZ data (is this necessary?)  ########
gc()
```

```
gzdata<-TSTATgetdata(TSTATname,"Whole","GZ");
gzdatavalues<-Vdecimate(gzdata[,2],decfactor);
gzdatatimes<-(Vdecimate(gzdata[,1],decfactor));
rm(gzdata);
gc();
plotxlim<-c(0,max(gzdatatimes));
plot(cbind(gzdatatimes,gzdatavalues),xlab="",ylab="",main="",
    col="black",type="l",xlim=plotxlim,ylim=c(10,140));

######## Always get Gsuit Pressure,if it is available ############
if (!is.null(eval(parse(text=paste("TSTATinfo$data$",insertion,
    "$Descr$data$GSP",sep="")))))
{##plot Gsuit Pressure
par(new=T);
gsdata<-TSTATgetdata(TSTATname,"Whole","GSP");
gsdatavalues<-Vdecimate(gsdata[,2],decfactor);
gsdatatimes<-(Vdecimate(gsdata[,1],decfactor));
rm(gsdata);
gc();
plot(cbind(gsdatatimes,gsdatavalues),col="blue",type="l",
    xlim=plotxlim,xlab="",ylab="",main="",yaxt="n");
}

######## Always get Mask Pressure, if it is available ############
if (!is.null(eval(parse(text=paste("TSTATinfo$data$",insertion,
    "$Descr$data$MSP",sep="")))))
{##plot Mask Pressure
par(new=T);
```

```
msdata<-callTSTAT(TSTATname,"getd","Whole","MSP");
msdatavalues<-Vdecimate(msdata[,2],decfactor);
msdatatimes<-(Vdecimate(msdata[,1],decfactor));
rm(msdata);
gc();
plot(cbind(msdatatimes,msdatavalues),col="green",type="l",
    xlim=plotxlim,xlab="",ylab="",main="",yaxt="n");
}
title(main=paste("Data Values in",ProfName,"Window (",insertion,")",
    sep=" "),xlab="Time (sec.)",ylab="GZ Value (degrees)")



  if(FunName=="Vintegratefull")
  {
   singlevalue<-c(ExtraVect[i,1],ExtraVect[i,2],ExtraVect[i,3]);
   #temp<-eval(parse(text=paste(FunName,"(datavalues,datatimes,
    singlevalue,...)",sep="")));
   temp<-Vintegratefull(datavalues,datatimes,singlevalue,...);
   result<-c(result,temp);
   par(new=T);
   ##plot(gzdatavalues,col="black",main=paste("Area Under",ChName,
   ##   "Curve in",ProfName,"Window (",insertion,")",sep=" "),
   ##   xlab="Sample Number (decimated by 10)",ylab="Area Under
   ##   Curve (Base Units*Samples",type="l");
   ## }

  if(FunName=="Gzpoints")
  {
```

```
 singlevalue<-ExtraVect[1];
 par(ask=F)
 #temp<-eval(parse(text=paste(FunName,"(datavalues,
 #    singlevalue,...)",sep="")));
 temp<-Gzpoints(datavalues,singlevalue,...);
 result<-c(result,temp);
 par(new=T,ask=T); # Prompt the user for this graph
 ##plot(gzdatavalues,col="black",main=
 ##   paste("GZ Transition Points (",insertion,
 ##   ")",sep=" "),xlab="Sample Number (decimated by 10)",ylab="",type="l");
}

if(FunName=="Gzpoints2")
{
 if(any(is.na(ExtraVect[i,]))) #If GZpoints didn't find a tilt
 {
  tag<-menu(c("GZ","GSP","MSP","SYS","HYS"),title=
  paste(insertion, ":  No HDT.  Choose reference TAG:",sep=""));
  tag<-c("GZ","GSP","MSP","SYS","HYS")[tag];

  rm(datavalues,datatimes,gzdatavalues,gzdatatimes);

  # Currently, the datatrac tags "GSP" and "MSP" are not available.
  if(tag == "GSP" || tag == "MSP")
  {
   stop("ERROR: Undefined datatrac tag.")
  }
```

```
data<-TSTATgetdata(TSTATname,ProfName,tag);
if (ChName != "SYS" && ChName != "DIA" && ChName !="ECG" &&
ChName != "ECGP")
{
 datavalues<-Vdecimate(data[,2],1);
 datatimes<-Vdecimate(data[,1],1);
}
else
{
 datavalues<-data[,2];
 datatimes<-data[,1];
}
rm(data);
gc()

par(new=F);

plot(cbind(datatimes,datavalues),col="black",main=paste(tag,"
Values in",ProfName,"Window (",insertion,")",sep=" "),
xlab="Time (sec.)",ylab="Value (Base Units)",type="l");
tempresult<-c()

#Prompt the User to select two points
for (k in 1:2)
{
cat("Select Point ",as.numeric(k),"\n");
temp<-identify(x=datatimes,y=datavalues,n=1,plot=F);
```

```
    abline(v=temp/100,col="red")
    tempresult<-c(tempresult,temp/100);
    }

##    if (tag=="GSP" || tag=="MSP")
##    {

   tempresult<-c(tempresult[1],tempresult[1],tempresult[2],tempresult[2]);
##    }
   result<-c(result,tempresult);
   }
   else result<-c(result,ExtraVect[i,]);
  }

  if(FunName=="verifyDFC")
  {
   DFCdata<-callTSTAT(TSTATname,"getDFC",ProfName,...);
   wholedata<-TSTATgetdata(TSTATname,"Whole",ChName);
   wholedata<-Vdecimate(wholedata[,2],10);

   starttime<-eval(parse(text=paste("TSTATinfo$data[[",i,"]]
    $Profiles$data$",ProfName,"$Start",sep="")));
   #return(datavalues);
   verifyDFC(DFCdata[[1]][i],datavalues,wholedata,starttime);
   par(new=T);
   ##plot(gzdatavalues,col="black",main=paste("DFC Data:",...,"in",
   ##   ProfName,"Window (",insertion,")",sep=" "),xlab=
   ##   "Sample Number (decimated by 10)",ylab="Data",type="l");
  }
 if(FunName=="calcDFC")

  {
   profiledata<-datavalues[datatimes>=(ExtraVect[i,1]) &
    datatimes<=(ExtraVect[i,2])];
   profiletimes<-datatimes[datatimes>=(ExtraVect[i,1]) &
    datatimes<=(ExtraVect[i,2])];
   if (length(profiledata)==0){ profiledata<-NA;profiletimes<-NA;}
   if (FunName=="slope")
   {
   temp<-lm(profiledata,profiletimes);
   }
   else
   {
   temp<-eval(parse(text=paste(...,"(profiledata)",sep="")));
   }

   ##xind<-which(as.logical(match(temp,profiledata,nomatch=0)));
   ##xind<-match(temp,profiledata,nomatch=0);
   ##return(temp);

   #Draw a vertical line at either end of the window
   abline(v=ExtraVect[i,1],col="red")
   abline(v=ExtraVect[i,2],col="red")

   par(new=F,ask=F);
   ##plot(gzdatavalues,col="black",main=paste("DFC Data:",...,"in",
##   ProfName,"Window (",insertion,")",sep=" "),
##   xlab="Sample Number (decimated by 10)",ylab="Data",type="l");

if (ChName == "rr" || ChName == "hr")
{
 ecgpdata<-TSTATgetdata(TSTATname,ProfName,"ECGP");
 ecgdata<-TSTATgetdata(TSTATname,ProfName,"ECG");
 plot(ecgpdata,xlab="",ylab="",col="red",xlim=plotxlim,
 ylim=c(-0.2,0.7),main="",yaxt="n");
 par(new=T);
 plot(ecgdata,xlab="",ylab="",type="l",col="blue",
 xlim=plotxlim,ylim=c(-0.2,0.7),main="",yaxt="n");
 par(new=T)
 plot(cbind(datatimes,datavalues),xlab="Time (s)",ylab="Channel Value",
 type="l",col="orchid",xlim=plotxlim,ylim=plotylim,
 main=paste(ChName," Data",sep=""));
 rm(ecgpdata,ecgdata)}
else
{
 plot(cbind(datatimes,datavalues),xlab="Time (s)",ylab="Channel Value",
 type="l",col="orchid",xlim=plotxlim,ylim=plotylim,
 main=paste(ChName," Data",sep=""));
}


###  Show Window and Calculated Value
abline(v=ExtraVect[i,1],col="red")
abline(v=ExtraVect[i,2],col="red")
```

```
    if (FunName == "slope")
    {
    abline(temp);
    }
    else
    {
    lines(c(ExtraVect[i,1],ExtraVect[i,2]),c(temp,temp),col="green")  ;
    }
    par(new=F,ask=T)
    ##plot(cbind(profiletimes[xind],profiledata[xind]),pch=16,col="red",
    ##   xlim=plotxlim,ylim=c(0,max(datavalues)));
    ##return(profiledata);
    ##return(c(profiletimes[xind],profiledata[xind]));
    par(ask=F) ##should be T
    result<-c(result,temp);

   }

if(FunName=="recalcDFC")

   {
    profiledata<-datavalues[datatimes>=(ExtraVect[i,1]) &
     datatimes<=(ExtraVect[i,2])];
    profiletimes<-datatimes[datatimes>=(ExtraVect[i,1]) &
     datatimes<=(ExtraVect[i,2])];
    if (length(profiledata)==0){ profiledata<-NA;profiletimes<-NA;}
```

```
##xind<-which(as.logical(match(temp,profiledata,nomatch=0)));
##xind<-match(temp,profiledata,nomatch=0);
##return(temp);

abline(v=ExtraVect[i,1],col="red")
abline(v=ExtraVect[i,2],col="red")

par(new=F,ask=F);
##plot(gzdatavalues,col="black",main=paste("DFC Data:",...,"in",
##   ProfName,"Window (",insertion,")",sep=" "),xlab=
##   "Sample Number (decimated by 10)",ylab="Data",type="l");

if (ChName == "rr" || ChName == "hr")
{
 ecgpdata<-TSTATgetdata(TSTATname,ProfName,"ECGP");
 ecgdata<-TSTATgetdata(TSTATname,ProfName,"ECG");
 plot(ecgpdata,xlab="",ylab="",col="red",xlim=plotxlim,
 ylim=c(-0.2,0.7),main="",yaxt="n");
 par(new=T);
 plot(ecgdata,xlab="",ylab="",type="l",col="blue",
 xlim=plotxlim,ylim=c(-0.2,0.7),main="",yaxt="n");
 par(new=T)
 plot(cbind(datatimes,datavalues),xlab="Time (s)",ylab="Channel Value",
 type="l",col="orchid",xlim=plotxlim,ylim=plotylim,
 main=paste(ChName," Data",sep=""));
 rm(ecgdata,ecgpdata)}
else
{
   plot(cbind(datatimes,datavalues),xlab="Time (s)",ylab="Channel Value",
   type="l",col="orchid",xlim=plotxlim,ylim=plotylim,
   main=paste(ChName, " Data",sep=""));
  }


  ###  Show Window and Calculated Value
  abline(v=ExtraVect[i,1],col="red")
  abline(v=ExtraVect[i,2],col="red")
  lines(c(ExtraVect[i,1],ExtraVect[i,2]),c(temp,temp),col="green")
  par(new=F,ask=T)
  ##plot(cbind(profiletimes[xind],profiledata[xind]),pch=16,col="red",
  ##   xlim=plotxlim,ylim=c(0,max(datavalues)));


   cat("Select Points to exclude from data (right click to end) \n");
   temp<-identify(profiletimes,profiledata);
   profiledata<-profiledata[-temp];
   profiletimes<-profiletimes[-temp];
   temp<-eval(parse(text=paste(...,"(profiledata)",sep="")));

   return(cbind(profiletimes,profiledata));

  }

 }
par(ask=T,mfrow=c(1,1),new=F);
```

```
# If the function we invoked was "Gzpoints" or "Gzpoints2", the output
# has to be in a specific matrix format.
if(FunName == "Gzpoints" || FunName == "Gzpoints2")
{
 result <- matrix(result,endnum,4,2)
}

result
}



callTSTAT<-function(ObjectName,FunName,Arg1="",Arg2="",Arg3="",...)
{
##This is a wrapper function to call any TSTAT function for the
##TSTAT object ObjectName which is a character string
##FunName is the TSTAT function (ie. getd, select, etc.) as a
##character string
##Arg1 , Arg2, etc. are arguments required by TSTAT as character strings
##Currently a max of four arguments are allowed but this could
##be easily extended
numargs<-nargs();

 if (numargs<2) {
     warning("callTSTAT requires at least TSTAT object name
         and function name as arguments");
     return()
 }
 if (numargs==2) callstring<-paste("temp<-",ObjectName,"$",
    FunName,"()",sep="");
if (numargs==3) callstring<-paste("temp<-",ObjectName,"$",
    FunName,"(Arg1)",sep="");
if (numargs==4) callstring<-paste("temp<-",ObjectName,"$",
    FunName,"(Arg1,Arg2)",sep="");
if (numargs==5) callstring<-paste("temp<-",ObjectName,"$",
    FunName,"(Arg1,Arg2,Arg3)",sep="");
if (numargs>5) {
    warning("callTSTAT only uses five arguments at this time");
    return()
}

temp<-eval(parse(text=callstring));
return(temp)

}

Vdecimate<-function(Vectorname, fact=10)
{
##This is used to decimate a vector of numbers by the factor fact
##VectorName is an actual vector, not a character string
##The routine works by pure vector manipulation, not using a for loop

count<-1;
Shortvector<-c();
reps<-floor(length(Vectorname)/fact);
remaining<-length(Vectorname)-reps*fact;
```

```
if(remaining!=0){
    indexvector<-c(rep(c(T,rep(F,fact-1)),reps),T,rep(F,remaining-1)) ;
}
else {indexvector<-rep(c(T,rep(F,fact-1)),reps) ;}
Shortvector<-Vectorname[indexvector];

return(Shortvector)
}



Vintegratefull<-function(Vname,Vtimes,ExtraVect,direction,delta=1)
{
##Vintegratefull is a function to numerically integrate a vector of data.
##The "full" refers to the fact that it can integrate below or
##above a given value (not both,ie. negative integrals
##don't cancel positive)
##Vname is a numeric vector, as is Vtimes, which contains
##corresponding timestamps
##Extravector is a numeric vector containing three numbers.
##The first is the time of the beginning of integration,
##the second is the end time.  The third value is the number
##to integrate above or below.
##Finally, indicate above or below through the direction argument.
##Acceptable values are "below" or "Below" otherwise above is assumed

 abline(v=ExtraVect[1],col="red")
 abline(v=ExtraVect[2],col="red")
 area<-0;
 profiledata<-Vname[Vtimes>=ExtraVect[1] & Vtimes<=ExtraVect[2]];
 if(length(profiledata)<2)return(NA);
 profiletimes<-Vtimes[Vtimes>=ExtraVect[1] & Vtimes<=ExtraVect[2]];

 plot(cbind(Vtimes,Vname),main="",xlab="",ylab="",type="l",col="blue");
 abline(v=ExtraVect[1],col="red")
 abline(v=ExtraVect[2],col="red")

 Horiz<-ExtraVect[3];
 if (direction=="below"||direction=="Below") cmp<- 1 else cmp<- -1;
 ##return(c( (ExtraVect[1]*10),(ExtraVect[2]*10)));
 for(i in 1:(length(profiledata)-1))
 {
  ##return(c(i,length(Vname)));
  ##return(1:length(profiledata)-1);
  if ((profiledata[i]-Horiz)*cmp>0) left<-0 else left<-profiledata[i]-Horiz;
  if ((profiledata[i+1]-Horiz)*cmp>0) right<-0 else
    right<-profiledata[i+1]-Horiz;
  y<-(left+right)/2;
  area<-area+y*delta;

##if((left!=0)||(right!=0))lines(c(i,i+1),c(y+Horiz,y+Horiz),col="red");
if((left!=0)||(right!=0))lines(c(profiletimes[i],profiletimes[i+1]),
    c(y+Horiz,y+Horiz),col="red");

##if((left!=0)||(right!=0)) lines(c(i+0.5,i+0.5),
##  c(y+Horiz,Horiz),col="red");
```

```
if((left!=0)||(right!=0)) lines(c(((profiletimes[i+1]-profiletimes[i])/2)+
    profiletimes[i],((profiletimes[i+1]-profiletimes[i])/2)+profiletimes[i]),
    c(y+Horiz,Horiz),col="red");
 }
abline(h=Horiz);
abs(area);
}



addfact<-function(DFname,lnames)
{
#addfact was written to make it easier to add a column
#of factors to a data frame.
##The whole dataframe (NOT its name as a  character string)
##is passed as DFname
##lnames is a CHARACTER VECTOR containing the levels of the factor
##For each row in the dataframe a menu showing the levels "lnames"
##is displayed and you are prompted to select one.
##The return value is a colomn of factors.  Note the original dataframe is
##not altered. Rather, the result must be cbinded to the dataframe manually

 numrows<-length(rownames(DFname));
 DFRowNames<-rownames(DFname);
 factors<-c();
 for (i in 1:numrows)
 {
  temp<-menu(lnames,title=paste("Select Factor for: ",DFRowNames[i]));
  factors<-c(factors,lnames[temp]);
 }
 factors<-factor(factors,levels=lnames);
 rm(numrows,DFRowNames,temp);
 factors;
}

addnumeric<-function(DFname,lnames)
{
#Similar to addfact. addnumeric was written to make it easier to add
#a column of numerics to a data frame from a finite numer of values.
##The whole dataframe (NOT its name as a  character string)
##is passed as DFname
##lnames is a NUMERIC VECTOR containing the levels of the variable
##For each row in the dataframe a menu showing the levels "lnames"
##is displayed and you are prompted to select one.
##The return value is a colomn of numbers.  Note the original dataframe
##is not altered. Rather, the result must be cbinded
##to the dataframe manually

 numrows<-length(rownames(DFname));
 DFRowNames<-rownames(DFname);
 columndata<-c();
 for (i in 1:numrows)
 {
  temp<-menu(lnames,title=paste("Select Value for: ",DFRowNames[i]));
  columndata<-c(columndata,lnames[temp]);
 }
 ##factors<-factor(factors,levels=lnames);
 rm(numrows,DFRowNames,temp);
 columndata;
}



Gzpoints<-function(Vname,delta=1,...)
{
##This function is written to locate the points on a Tilt Table profile
##Currently, four points are located (roughly).  A start
##from Head Up tilt is assumed.
##The points found are:
##  Start of transition from HU to HD
##  Time of reaching HD
##  Start of transition from HD to HU
##  Time of reaching HU
##The returned value is a vector containing the four times.
##If no tilt was found, the vector contains na's
 NotHUT<-Vname>18;
 HDT<-Vname>130;
 ##return(HDT)

 if (any(HDT))
 {
  for (i in 1:length(Vname))
  {
   if(NotHUT[i]==T && NotHUT[i+1]==T && NotHUT[i+2]==T && NotHUT[i+3]==T)
            {
    pt1<-i;
 break;   ##found start of HDT transition

       }
}
for (i in pt1:length(Vname))
{
 if(NotHUT[i]==F && NotHUT[i-1]==T && NotHUT[i-2]==T && NotHUT[i-3]==T)
       {
 pt4<-i;
 break;   ##found start of HDT transition

       }
}
for (i in 1:length(Vname))
{
 if(HDT[i]==T && HDT[i+1]==T && HDT[i+2]==T && HDT[i+3]==T)
       {
 pt2<-i;
 break;   ##found start of HDT transition

       }
}
for (i in pt2:length(Vname))
{
 if(HDT[i]==F && HDT[i-1]==T && HDT[i-2]==T && HDT[i-3]==T)
       {
 pt3<-i;
 break;   ##found start of HDT transition
```

```
            }

  }
 }
 else
 {

  pt1<-NA;
  pt2<-NA;
  pt3<-NA;
  pt4<-NA;
 }
 plot(Vname,main="",xlab="",ylab="",type="l",col="black");
 abline(v=pt1,col="red")
 abline(v=pt2,col="purple")
 abline(v=pt3,col="blue")
 abline(v=pt4,col="green")
 c(pt1,pt2,pt3,pt4)/(10^delta)

}

verifyDFC<-function(DFCdata,Profiledata,Wholedata,starttime)
{
##This function is usd to verify the DFC data which may have
##been obtained directly by TSTAT
##However, TSTAT doesn't offer visual verification of the value obtained
##DFCdata is a vector of data obtained from the TSTAT command $getDFC
```

```
##Profiledata is the data from the window in which the DFC
##data should be calculated
##Wholedata is the entire signal of the channel
##starttime is the starting time of the window
##This function plots the whole data signal, overlays the data in the
##profile window, and shows the TSTAT-calculated value
##as a red horizontal line
##This function only plots the results for one insertion so, in general,
##it will be called in a loop for all insertions

##documentation for this will be completed later...
 ##verifyDFC<-function(FunName,TSTATname,ProfName,ChName,DFCName,...)
 par(new=F);
 plot(Wholedata,ylim=range(Wholedata),main="",xlab="",
    ylab="",type="l",col="blue");
 xlength<-length(Wholedata);
 par(new=T);


plot((starttime*10):(starttime*10+length(Profiledata)-1),Profiledata,
    col="red",ylim=range(Wholedata),xlim=c(1,xlength),main="",
    xlab="",ylab="",type="b",pch=16);
 abline(h=DFCdata,col="red");
 ##par(new=F);


 ##verifyDFC<-function(DataValues,SingleValue,DFCName
 ##{
```

```
##par(ask=T);
##temp<-callTSTAT(TSTATname,"getDFC",ProfName,DFCName);

##for (i in 1:length(temp[[1]]))
##  {
##  par(new=F);
##data<-callTSTAT(TSTATname,"getd",ProfName,ChName);
##data<-Vdecimate(data[,2],10);
##plot(data);
##abline(h=temp[[1]][i]);
##par(new=T);
##  }
##par(ask=F);
}

##TexCurFig, TexSimpleAOV, TexRMAOV, TexSummaryAOV, TexTable, Texify
##TexStartTableFig, TexEndTableFig, TexFindPSNum, TexTableFig

##TexCurFig<-function(TexFileName)
TexCurFig<-function(TexFileName,caption,shortcaption,scalefactor,leftshift)
{
FigFileNum<-1;
FoundNextFile<-F;
while(FoundNextFile==F)
  {
if(!file.exists(file.path(paste(TexFileName,".",
    as.character(FigFileNum),".ps",sep=""))))
    {
```

```
    FoundNextFile<-T;

    FigFileName<-paste(TexFileName,".",as.character(FigFileNum),".ps",sep="");

postscript(file.path(FigFileName));
dev.set(dev.prev())
dev.copy(which=dev.next());
dev.off();
##dev.print(device=postscript,file=file.path(FigFileName));
##temp<-dev.cur();
    break;
    }
FigFileNum<-FigFileNum+1;

  }

##outline<-paste("\\includegraphics[angle=270,
##  width=\\plotWidth]{",FigFileName,"}","\\\\",sep="");
outline<-paste("\\fig{",FigFileName,"}{",caption,"}
    {",shortcaption,"}{",scalefactor,"}{",leftshift,"}",
    sep="");
write(outline,file=file.path(paste(TexFileName,".tex",sep="")),append=T);
FigFileName

}


TexSummaryAOV <- function (TexFileName,object,...)
```

```
{
if(length(summary(object))==1)TexSimpleAOV(TexFileName,object);
if(length(summary(object))>1)TexRMAOV(TexFileName,object);
}



TexSimpleAOV <- function (TexFileName,object,...)
{

##TexTable<-function(TexFileName,VectorData,Numcols)




##temp<-dim(VectorData);

summ<-summary(object);
##Numstratum<-length(summ);
##StratumNames<-names(object)[-1];

Numcols<-dim(summ[[1]])[2];

##RowLabels<-c();
##for (i in 1:Numstratum)
##{
RowLabels<-rownames(summ[[1]]);
##}

ColLabels<-c("",gsub(">","$>$",names(summ[[1]])));
```

```
##outtext<-paste("\\begin{tabular}{",rep("p{1.5cm}",Numcols),"}",sep="")
##if(Numstratum>1) outtext<-paste("\\begin{tabular}
##  {p{2cm}p{2cm}",paste(rep("p{1.5cm}",Numcols),
##  collapse=""),"}"," \\hline",sep="");

outtext<-paste("\\begin{tabular}[H]{p{2cm}",paste(rep("p{1.5cm}",Numcols),
    collapse=""),"}"," \\hline",sep="");
tabledata<-c();



##Titles
for(j in 1:(Numcols+1))
   {
  ## if(j==0)tabledata<-paste(tabledata,as.character(ColLabels[j]),sep="");
   if(j==1)tabledata<-paste(tabledata,as.character(ColLabels[j]),sep="");
   if((j<=Numcols+1 && j>1) || (j==Numcols && Numcols==2))
    tabledata<-paste(tabledata,as.character(ColLabels[j]),sep=" & ");
  ## if((j==Numcols+1) && (Numcols !=2))
  ##    tabledata<-paste(tabledata,as.character(ColLabels[j]),sep=" & ");
   }
  tabledata<-paste(tabledata," \\\\","\\hline ",sep="");
##  tabledata<-paste(tabledata," \\\\",sep="");

##Stratum
##for (k in 1:Numstratum)

##{
```

```
Numrows<-dim(summ[[1]])[1];
 for(i in 1:Numrows)
 {
   for(j in 0:(Numcols))
   {
##   if(j==-1 && i==1)tabledata<-paste(tabledata,StratumNames[k],sep="");
##   if(j==-1 && i!=1)tabledata<-paste(tabledata,"",sep="");
   if(j==0)tabledata<-paste(tabledata,as.character(
    rownames(summ[[1]])[i]),sep="");



if(j>=1) tabledata<-paste(tabledata,as.character(]
    signif(summ[[1]][i,j],digits=4)),sep=" & ");
##if((j<Numcols && j>1) || (j==Numcols && Numcols==2))
##  tabledata<-paste(tabledata,as.character(
##  signif(summ[[k]][[1]][i,j],digits=4)),sep=" & ");
##if((j==Numcols) && (Numcols !=2))
##  tabledata<-paste(tabledata,as.character(
##  signif(summ[[k]][[1]][i,j],digits=4)),sep=" & ");

   }



##  tabledata<-paste(tabledata,as.character(
##  VectorData[i,Numcols]),sep=" ");
##  tabledata<-paste(tabledata," \\\\","\\hline ",sep="");
    tabledata<-paste(tabledata," \\\\",sep="");
```

```
## }

  tabledata<-paste(tabledata," \\\\","\\hline ",sep="");

}

##return(tabledata);
outtext<-paste(outtext,tabledata,"\\end{tabular} \\\\" ,sep=" ");
write(outtext,file=file.path(paste(TexFileName,".tex",sep="")),append=T);
##FigFileName
return(tabledata)
}


TexRMAOV <- function (TexFileName,object,...)
{

##TexTable<-function(TexFileName,VectorData,Numcols)



##temp<-dim(VectorData);

summ<-summary(object);
Numstratum<-length(summ);
StratumNames<-names(object)[-1];

Numcols<-dim(summ[[1]][[1]])[2];
```

```
RowLabels<-c();
for (i in 1:Numstratum)
{
RowLabels<-c(RowLabels,rownames(summ[[i]][[1]]));
}

ColLabels<-c("","",gsub(">","$>$",names(summ[[1]][[1]])));

##outtext<-paste("\\begin{tabular}{",rep("p{1.5cm}",Numcols),"}",sep="")
if(Numstratum>1) outtext<-paste("\\begin{tabular}[H]{p{2cm}p{2cm}",
    paste(rep("p{1.5cm}",Numcols),collapse=""),"}"," \\hline",sep="");
##if(Numstratum==1) outtext<-paste("\\begin{tabular}{p{2cm}",
##  paste(rep("p{1.5cm}",Numcols),collapse=""),"}"," \\hline",sep="");
tabledata<-c();


##Titles
for(j in 1:(Numcols+2))
   {
  ## if(j==0)tabledata<-paste(tabledata,as.character(ColLabels[j]),sep="");
   if(j==1)tabledata<-paste(tabledata,as.character(ColLabels[j]),sep="");
   if((j<=Numcols+2 && j>1) || (j==Numcols && Numcols==2))
    tabledata<-paste(tabledata,as.character(ColLabels[j]),sep=" & ");
  ## if((j==Numcols+1) && (Numcols !=2))
  ##    tabledata<-paste(tabledata,as.character(ColLabels[j]),sep=" & ");
   }
  tabledata<-paste(tabledata," \\\\","\\hline ",sep="");
```

```
##  tabledata<-paste(tabledata," \\\\",sep="");

##Stratum
for (k in 1:Numstratum)

{
Numrows<-dim(summ[[k]][[1]])[1];
 for(i in 1:Numrows)
 {
   for(j in -1:(Numcols))
   {
   if(j==-1 && i==1)tabledata<-paste(tabledata,StratumNames[k],sep="");
   if(j==-1 && i!=1)tabledata<-paste(tabledata,"",sep="");
   if(j==0)tabledata<-paste(tabledata,as.character(
    rownames(summ[[k]][[1]])[i]),sep=" & ");


if(j>=1) tabledata<-paste(tabledata,as.character(
    signif(summ[[k]][[1]][i,j],digits=4)),sep=" & ");
##if((j<Numcols && j>1) || (j==Numcols && Numcols==2))
##  tabledata<-paste(tabledata,as.character(
##  signif(summ[[k]][[1]][i,j],digits=4)),sep=" & ");
##if((j==Numcols) && (Numcols !=2))
##  tabledata<-paste(tabledata,as.character(
##  signif(summ[[k]][[1]][i,j],digits=4)),sep=" & ");

   }
##  tabledata<-paste(tabledata,as.character(
##  VectorData[i,Numcols]),sep=" ");
##  tabledata<-paste(tabledata," \\\\","\\hline ",sep="");
    tabledata<-paste(tabledata," \\\\",sep="");

 }

  tabledata<-paste(tabledata," \\\\","\\hline ",sep="");

}

##return(tabledata);
outtext<-paste(outtext,tabledata,"\\end{tabular} \\\\" ,sep=" ");
write(outtext,file=file.path(paste(TexFileName,".tex",sep="")),
    append=T);
##FigFileName
return(tabledata)
}


TexTable<-function(TexFileName,VectorData,Numcols)
{

temp<-dim(VectorData);
Numrows<-temp[1];
Numcols<-temp[2];
RowLabels<-rownames(VectorData);
```

```
ColNames<-c("",names(VectorData));

##outtext<-paste("\\begin{tabular}{",rep("|p{1cm}",
##  Numcols),"|}",sep="")
outtext<-paste("\\begin{tabular}{",paste(rep("|p{1cm}",Numcols+1),
    collapse=""),"|}"," \\hline" ,sep="");
tabledata<-c();

##Titles
for(j in 1:(Numcols+1))
   {
  ## if(j==0)tabledata<-paste(tabledata,as.character(ColNames[j]),sep="");
   if(j==1)tabledata<-paste(tabledata,as.character(ColNames[j]),sep="");
   if((j<=Numcols+1 && j>1) || (j==Numcols && Numcols==2))
    tabledata<-paste(tabledata,as.character(ColNames[j]),sep=" & ");
  ## if((j==Numcols+1) && (Numcols !=2))
  ##    tabledata<-paste(tabledata,as.character(ColNames[j]),sep=" & ");
   }
  tabledata<-paste(tabledata," \\\\","\\hline  ",sep="");

for(i in 1:Numcols)
{
if(is.numeric(VectorData[1,i]))
    VectorData[,i]<-signif(VectorData[,i],digits=4);


}
```

```
 for(i in 1:Numrows)
 {
   for(j in 0:(Numcols+1))
   {
   if(j==0)tabledata<-paste(tabledata,as.character(RowLabels[i]),sep="");
   if(j==1)tabledata<-paste(tabledata,as.character(
    VectorData[i,j]),sep=" & ");
   if((j<Numcols && j>1) || (j==Numcols &&
    Numcols==2))tabledata<-paste(tabledata,
    as.character(VectorData[i,j]),sep=" & ");
   if((j==Numcols) && (Numcols
!=2))tabledata<-paste(tabledata,as.character(VectorData[i,j]),sep=" & ");

   }

##   tabledata<-paste(tabledata,as.character(
##  VectorData[i,Numcols]),sep=" ");
  tabledata<-paste(tabledata," \\\\","\\hline ",sep="");

}

##return(tabledata);
outtext<-paste(outtext,tabledata,"\\end{tabular} \\\\" ,sep=" ");
write(outtext,file=file.path(paste(TexFileName,".tex",sep="")),append=T);
##FigFileName
return(tabledata)
}
```

```
Texify <-function(FileName,...)
{
cmdline<-paste("cat texheader.tex ",FileName,".tex texend.tex > ",
    FileName,"full.tex",sep="");
system(cmdline);

cmdline<-paste("latex ", FileName,"full.tex",sep="");
system(cmdline);

cmdline<-paste("latex ", FileName,"full.tex",sep="");
system(cmdline);

cmdline<-paste("xdvi ", FileName,"full&",sep="");
system(cmdline);

##return(cmdline);
}

##TexStartTableFig, TexEndTableFig, TexTableFig, TexCurFig

TexStartTableFig<-function(TexFileName,numCols,TableTitle)
{
outline<-paste("\\begin{showPlots}{",numCols,"}{",TableTitle,"}",sep="");
write(outline,file=file.path(paste(TexFileName,".tex",sep="")),append=T);
TexFileName
}

TexEndTableFig<-function(TexFileName)
```

```
{
outline<-paste("\\end{showPlots}",sep="");
write(outline,file=file.path(paste(TexFileName,".tex",sep="")),append=T);
TexFileName
}

TexTableFig<-function(TexFileName,caption)
{
##generate plot
FigFileNum<-TexFindPSNum(TexFileName);
FigFileName<-paste(TexFileName,".",
    as.character(FigFileNum),".ps",sep="");

postscript(file.path(FigFileName));
dev.set(dev.prev())
dev.copy(which=dev.next());
dev.off();
##dev.print(device=postscript,file=file.path(FigFileName));
##temp<-dev.cur();


##outline<-paste("\\includegraphics[angle=270,width=\\plotWidth]
##  {",FigFileName,"}","\\\\",sep="");
outline<-paste("\\nextPlot{",FigFileName,"}{",caption,"}",sep="");
write(outline,file=file.path(paste(TexFileName,".tex",sep="")),append=T);
FigFileName
}

TexFindPSNum<-function(TexFileName)
{
##routine to look for next available ps file number for
##a given LaTeX Document
FigFileNum<-1;
FoundNextFile<-F;
while(FoundNextFile==F)
  {
if(!file.exists(file.path(paste(TexFileName,".",
    as.character(FigFileNum),".ps",sep=""))))
    {
    FoundNextFile<-T;

    FigFileName<-paste(TexFileName,".",
        as.character(FigFileNum),".ps",sep="");
    }
else FigFileNum<-FigFileNum+1;
  }
return(FigFileNum)

}

##TexCurFig<-function(TexFileName)
#TexCurFig<-function(TexFileName,caption,shortcaption,
##  scalefactor,leftshift)
```

```
#{
#
#FigFileNum<-TexFindPSNum(TexFileName);
#
#FigFileName<-paste(TexFileName,".",as.character(FigFileNum),
#   ".ps",sep="");
#
#postscript(file.path(FigFileName));
#dev.set(dev.prev())
#dev.copy(which=dev.next());
#dev.off();
###dev.print(device=postscript,file=file.path(FigFileName));
###temp<-dev.cur();
##outline<-paste("\\includegraphics[angle=270,width=\\plotWidth]
##  {",FigFileName,"}","\\\\",sep="");
#outline<-paste("\\fig{",FigFileName,"}{",caption,"}{",shortcaption,"}
#   {",scalefactor,"}{",leftshift,"}",sep="");
#write(outline,file=file.path(paste(TexFileName,".tex",sep="")),append=T);
#FigFileName
#
#}


#
# Multivariate Analysis of Repeated Measures Data.
#
# So far, the analysis of repeated measures data has been
# restriced to the univariate problems
# (i.e. single response variable, or a single variable of interest).
# Here, we wish to include functions capable of analyzing multivariate
# problems, where we might have to look at a number of different
# variables of interest.
#
# Note: the following techniques to be described are often
# analogous to the single response variable case.

# multrep: ESIstat procedure for analyzing multivariate
# repeated measures problems.
# This function computes test statistic and the p-values of the
# null hypothesis that all means are equal given that they are
# all correlated equally (applicable for repeated measures problem),
# and gives the 95% confidence interval.
multrep <- function(DF, datacol, bycol, subsetval)
{
    # Select the desired subsets.
    temp1 <- P239subset(DF,bycol,subsetval[1])
    temp1 <- temp1[,datacol]
    temp2 <- P239subset(DF,bycol,subsetval[2])
    temp2 <- temp2[,datacol]

    # combine the two sets of data as a major matrix.
    matr <- cbind(temp1,temp2)

    # compute the sample variance of the matrix.
    ss <- var(matr)

    p <- 2
```

```
ones <- matrix(rep(1,p),p,1)

# now calculate the ANOVA table using vector notation.
xbar <- matrix(apply(matr,2,mean),p,1)

n <- length(temp1)

vv <- (n-1)*ss
v1 <- n * (t(xbar) %*% xbar - (t(ones) %*% xbar)^2/p)
v2 <- (t(ones) %*% vv %*% ones) / p
v <- sum(diag(vv)) - v2

# Calculate the test statistic.
test.stat <- ((n-1)*v1) / v

# Now calculate the p-value.
pvalue <- 1-pf(test.stat,p-1,(n-1)*(p-1))

# Compute the 95% confidence interval.
# There are 2 types of confidence intervals: Scheffe and Tukey.
# Whichever has the shorter confidence interval is the
# one we should use.
#
# First, compute the Scheffe C.I.
cc <- qf(.95,p-1,(n-1)*(p-1))
aa <- matrix(c(-1,1),2,1)
std.err.scheffe <- sqrt((cc*(t(aa)%*%aa)*v)/((n-1)*n))
```

```
    # Now compute the Tukey C.I.
    c1 <- qtukey(.95,p,(n-1)*(p-1))
    std.err.tukey <- .5*c1*sqrt(v/((n-1)*n*(p-1))) * sum(abs(aa))

    if(std.err.scheffe <= std.err.tukey)
    {
        conf.int <- c((t(aa)%*%xbar) - std.err.scheffe,
            (t(aa)%*%xbar) + std.err.scheffe)
    } else {
        conf.int <- c((t(aa)%*%xbar) - std.err.tukey,
            (t(aa)%*%xbar) + std.err.tukey)
    }

    return(cat("The test statistic, p-value and the confidence
        interval of the hypothesis test is:\n"),
        test.stat, pvalue,conf.int)
}

# multTest: ESIstat procedure, multivariate equivalent of a
# paired t-test.
# This function computes test statistic and the p-values of
# the null hypothesis that all means
# are equal, given that the data are not independently distributed.
multTest <- function(DF, datacol, bycol, subsetval)
{
        # Select the desired subsets.
        temp1 <- P239subset(DF,bycol,subsetval[1])
        temp1 <- temp1[,datacol]
    temp2 <- P239subset(DF,bycol,subsetval[2])
    temp2 <- temp2[,datacol]

    p <- length(datacol); n <- dim(temp1)[1]
    # Compute the sample means for each of the data sets.
    xbar <- apply(temp1,2,mean);ybar <- apply(temp2,2,mean)
    xbar <- matrix(xbar,p,1);ybar <- matrix(ybar,p,1)

    # Now compute the difference between the two means.
    diff.mn <- xbar-ybar

    # Now compute the covariance of the difference between
# the two data and its inverse.
    diff <- temp1-temp2
    diff.var <- var(diff)
sinv <- solve(diff.var)

    # Calculate the test statistic.
    f <- n-1
    test.stat <- (t(diff.mn) %*% sinv %*% diff.mn) *
    ((n*(f-p+1))/(f*p))

    # Calculate the p-value.
    pvalue <- 1-pf(test.stat,p,f-p+1)

    # Now calculate the 95% confidence intervals.
    Talpha <- sqrt(((f*p)/(f-p+1)) * qf(.95,p,f-p+1))
```

```
        # Compute the Bonferroni interval
        talpha <- qt(1-(.05/(2*p)),f)

        conf.int <- matrix(0,p,2)
        if(Talpha >= talpha)
        {
                for(i in 1:p)
                {
                        conf.int[i,] <- c(diff.mn[i] -
                (Talpha*sqrt(diff.var[i,i]/n)),diff.mn[i] +
                (Talpha*sqrt(diff.var[i,i]/n)))
                }
        } else  {
                for(i in 1:p)
                {
                        conf.int[i,] <- c(diff.mn[i] -
                (talpha*sqrt(diff.var[i,i]/n)),diff.mn[i] +
                (talpha*sqrt(diff.var[i,i]/n)))
                }
        }

        return(cat("The test statistic, pvalue and 95%
        confidence interval is:\n"),test.stat,pvalue,conf.int)
}

CreatePaper<-function(name,TexFile,type){

#Private Functions
```

```
#  print.PaperObj<-function(){
#   cat("This is the print function for class PaperObj");
#  }

  print.Paper<-function(object){
    #print.default(object$Title);
    cat("Title: ",object$Title, "\n");
    #print.default(object$filename);
    cat("File: ",object$filename, "\n");
    cat("Type: ",object$papertype, "\n");

    if(length(object$sections)>0){
#   cat("printing sections\n");
     for (i in 1: length(object$sections)){
      print.section(object,i);
     }
    }
    else{ cat("Sections: None\n")};

  #print.default(object$papertype);
  #print.sections(object$sections);
  }

  print.section<-function(object,sectionnum){
    #print.default(names(object$sections)[sectionnum]);
    cat(names(object$sections)[sectionnum], "\n");
#   print.default(paste("Title: ",
#       object$sections[[sectionnum]]$Title, sep="") );
```

```
  cat("  Title: ",object$sections[[sectionnum]]$Title, "\n");
# for (i in 1:length(object$sections[[sectionnum]])){
  cat("  print.analysis method goes here\n");
# }

 }

 print.subsections<-function(subsections){
  for (i in 1:length(names(subsections))){
  cat("this would be a subsection");
  }

 }

# basedir<-c("d:\\LinuxWork\\Papers\\");
 basedir<-c("/mnt/D/LinuxWork/Papers/")

 newfile<-file.path(paste(basedir,TexFile,".tex",sep=""));
 if(file.exists(newfile)) cat("File Already Exists\n")
 else { #Create a new file
  file.create(newfile);
  outtext<-c("%ESI PaperObj%Beginning-
    Do not edit lines beginning with %ESI PaperObj%\\\\")
  write(outtext,file=newfile,append=T);
  cat("Created File: ", newfile, "\n",sep="");
  TexFile<-newfile;

 }
```

```
  collection<-function(objectname,filepath,Ptype){
    x<-list(Title=objectname,filename=filepath,
    sections=list(),papertype=Ptype);
    class(x)<-c("collection");  #????Purpose
    return(x);
  }

  new.section<-function(SectionName){

    newSection<-list(Title=SectionName,Analyses=list());
    class(newSection)<-c("section");
    return(newSection);

  }

  add.section<-function(object,sectname,sectnum,sectlevel=1){
   if(!file.exists(TexFile)) {cat("File Doesn't Exist\n");return()}
   if(is.null(sectnum)){sectnum<-length(object$sections)+1}
   if(is.null(sectlevel)){sectlevel<-1;}

 #   if(sectnum > (length(object$sections)+1))
 #  {cat("Section Number too high");return()}
    if(sectnum<length(object$sections)){
      cat("Warning, section number too low. A section may be overwritten!");
      }
      #Add Section
#      object$sections[[sectnum]]<-list(Title=sectname);
#   Title is the title which appears in the Tex File
```

```
      object$sections[[sectnum]]<-new.section(sectname);
      #Title is the title which appears in the Tex File


    #return(object$sections[[sectnum]]);
    tempnames<-names(object$sections);
    tempnames[sectnum]<-paste("Sect",sectnum,sep="");
    names(object$sections)<-tempnames;
    #This is the title used by R

    outtext<-paste("%ESI PaperObj%-Sec",sectnum,
        "Do not edit this line\\\\",sep="")
    ##Need to locate proper place here
    write(outtext,file=TexFile,append=T);  #Comment the line in the file
    prefix<-paste(rep("sub",(sectlevel-1)),sep="",collapse="");
    outtext<-paste("\\",prefix,"section{",sectname,"}",sep="");
    write(outtext,file=TexFile,append=T);   #TeX code



   return(object);


  }
x<-collection(name,TexFile,type);
# x is the instance variable which cannot be directly
#accessed without class methods
class(x)<-c("PaperObj");
```

```
#Public Functions
self<-list(
  print=function(){
#    print.PaperObj(x);
   cat("This is the print function for class PaperObj\n");
   print.Paper(x);
   return(invisible())
  }
  ,get=function(attrib){
   cat("This is the get function for class PaperObj\n");
   if(missing(attrib)) return(names(x))
   else eval(parse(text=paste("x$",attrib,sep="")));



  }
  ,addsection=function(sectname,sectnum=NULL,sectlevel=NULL){
   cat("This is the addsection function for class PaperObj\n");
   if(missing(sectname)){cat("You must enter a section");return()}
   else{
    x<<-add.section(x,sectname,sectnum,sectlevel);
   }
##  ,newpaper=function()

   }
  ,returnobject=function(){
    return(x)
    }

   ,manobject=function(newobject){
     x<-newobject;
    }

);

class(self)<-c("PaperObj",class(self))
return(invisible(self))
}

print.PaperObj<-function(x,...){
#cat("temporary message");
x$print();
}

FindSection<-function(sectnum){
searchprefix<-c("%ESI PaperObj%");

}
```

Appendix B:

ESIstat R Documentation

R documentation

of all in ‘man’

September 28, 2001

`CreatePaper` *Creates Paper Object* `CreatePaper`

Description

This function allows the user to create a Paper Object. Encapsulated within this object are private functions which are intended to perform the low-level manipulations of LaTeX files, transparent to the user, as well as public functions (e.g. inserting paragraphs, figures or tables) which can be called from outside the object

Usage

```
CreatePaper(name,TexFile,type)
```

Value

An object with defined sections.

Examples

```
#Create an object called "Junk" to generate the LaTeX file "junk.tex".
junk <- CreatePaper("Junk Paper", "junk","Test")
```

`Dfapplyfun` *Apply special functions to all insertions in a TSTAT object* `Dfapplyfun`

Description

This is a generic sort of function to loop through the insertions in a TSTAT object The returned vector of data contains the same number of rows (in the same order) as the TSTAT object. It is used mainly to loop through the insertions in a TSTAT object and apply specialized functions

Usage

```
Dfapplyfun(FunName,ExtraVect,TSTATname,ProfName,ChName,startnum,endnum,...)
```

Arguments

`FunName`	The name of the function to apply for each insertion. Although any functon can potentially be called, a few certain ones are specifically designed to be used with Dfapplyfun They are: Vintegratefull, Gzpoints, Gzpoints2, calcDFC, verifyDFC, and recalcDFC. Refer to those functions for more info on how to use them separately
`ExtraVect`	An extra vector (or array) to be used for communicating with subfunctions For the functions mentioned under FunName, this argument has special meanings ..
`TSTATname`	A string containing the name of a valid TSTAT object
`ProfName`	A string containing the name of a valid profile within TSTATobject
`ChName`	A string containing the name of a valid Datatrac file TAG or any string which is handled by TSTATgetdata. Refer to TSTATgetdata for valid strings
`startnum`	(optional) The number of the TSTAT insertion to begin looping through.If unspecified, all insertions are processed.
`endnum`	(optional) The number of the TSTAT insertion to end looping through. If unspecified, all insertions are processed
`...`	Further arguments to be passed along to subfunctions. In particular, Vintegrate full expects to be passed a string for the direction of integration, and verifyDFC expects a string containing a valid DFC name in the TSTATobject

Value

A vector containing the results of the function applied to each insertion of the TSTAT object. In addition, for each insertion, a plot is produced to display the results of the calculation

Examples

```
#mattsobj is a TSTAT object with 6 insertions and a window titled
#"Whole" which has been defined to contain the entire signal.
# Also, a TSTAT DFC has been added to calculate the minimum of
#systolic blood pressure, titled "MinBP"

GZtimes<-Dfapplyfun("Gzpoints",1,"mattsobj","Whole","GZ",1,6)
#Locate GZ transitions in the GZ signal for insertions in mattsobj
#using the window "Whole", for insertions 1 to 6.  Store result in
#GZtimes

GZtimes<-Dfapplyfun("Gzpoints2",GZtimes,"mattsobj","Whole","GZ",1,6)
#Loop through GZtimes to find insertions where GZ transitions were
#missed on last pass

meanBPpre<-Dfapplyfun("calcDFC",GZtimes[,1]-5,"mattsobj","Whole","HYS",1,6,"mean")
#for insertions 1 to 6 in mattsobj, calculate the mean of systolic
#blood pressure (datatrac Tag = HYS) from the from 5 seconds before
#start of tilt to start of tilt

integralBPpre<-Dfapplyfun("Vintegratefull", cbind(GZtimes[,3],GZtimes[,3]+10,100)
  ,"mattsobj","Whole","HYS",1,6,"below")
#for insertions 1 to 6 in mattsobj calculate the integral of blood
#pressure below the value of 100, starting at the beginning of
#head-down to head-up transition and lasting for 10 seconds

Dfapplyfun("verifyDFC",NULL,"mattsobj","Whole","HYS",1,6,"MinBP")
#for insertions 1 to 6 in mattsobj loop through the insertions, plotting
#the data and the DFC values TSTAT calculated.
```

`Gzpoints` *Function to extract key time points from a GZ profile* `Gzpoints`

Description

This function was written to locate the points on a Tilt Table profile. Currently, four points are located (roughly). A start from Head Up tilt is assumed and the tilt levels assumed are HU (15 degrees) and HD (135 degrees). The points found are: Start of transition from HU to HD, Time of reaching HD, Start of transition from HD to HU, Time of reaching HU. This function would have to be altered for more complex tilt profiles, or for different HU/HD levels

Usage

```
Gzpoints(Vname)
```

Arguments

`Vname`	A vector of numerics which contains a tilt profile from head up tilt to head down tilt, and back to head up tilt.

Value

A vector of numerics of length four containing the four times. If no tilt is found, a vector of length for containing NA's.

Examples

```
GZtransitions<-Gzpoints(Tiltdata)        #for a vector represnting a tilt
                                          #signal locate four transition times.
```

`P239.t.test` *Perform t test on two groups of data from a dataframe* `P239.t.test`

Description

This function performs a t.test on two groups of data from a data frame. The column to be used for extracting groups is bycol. The actual data to be tested is datacol. To select the desired groups, pass the vector of factors in "subsetval".

Usage

```
P239.t.test(DF,datacol,bycol,subsetval)
```

Arguments

`DF`	The dataframe containing the data and the groups (factors)
`datacol`	A numeric indicating the column number of DF which contains the data.
`bycol`	A numeric indicating the column number of DF which contains the group names (factors)
`subsetval`	A vector containing two strings, representing the names of groups to compare.

Value

An object of type t.test

Examples

```
#mattsDF is a Data frame with numeric values in column 1 and factors in column 5

P239.t.test(mattsDF,1,5,c("fact1","fact2"))
#Perform a t.test on the data in column #1, using the factors in column
#5  Compare between "fact1" and "fact2"
```

`P239barplot` *Produce a series of barplots for each subject and output to TeX File* `P239barplot`

Description

This function produces a series of barplots for each subject in "subjectcol" and outputs the results to a Tex file as a table of figures. Currently, the function produces one plot for each of columns 3,6,9,12 and uses the factors in column 14 as the groupings (x axis of barplot). This could be easily generalized

Usage

```
P239barplot(FileName,DF,subjectcol)
```

Arguments

`FileName`	A string containing the name of the TeX file to be produced, excluding the extention
`DF`	The dataframe containing the data to be plotted. The data is plotted from columns 3,6,9,12, against the factors in column 14. This will be generalized.
`subjectcol`	The column containing the names of subjects for which each will have four plots produced.

Value

None. The series of plots is appended to the specified TeX file as a table of figures.

Examples

```
#DF is a Data frame with numeric values in column 1 and factors in column 5

P239barplot("/mnt/D/temp/test",DF,2)     #Create a boxplot of the data
                                          #where the subjects appear in column 2
```

`P239boxplot` *Prepare Boxplots from a dataframe* `P239boxplot`

Description

This function is used to prepare colored boxplots of a DataFrame (DF). The Dep input indicates the Dependant variable to be plotted on the y axis. The Indep input indicates the Independant variable to be plotted along the x axis. The plot parameters are read beforehand and reset afterwards.

Usage

```
P239boxplot(DF,Dep,Indep,logvect=rep(T,26))
```

Arguments

`DF`	The Dataframe containing the data to be plotted
`Dep`	The number of the column to be treated as the dependent variable
`Indep`	The number of the column to be treated as the independent variable
`logvect`	A logical vector which is ised to select the colors of the boxes. So far this was written specifically to handle the 26 experiments in the P239 experiment and still has to be generalized

Value

None. A boxplot is produced and displayed.

Examples

```
#mattsDF is a Data frame with numeric values in column 1 and factors in column 5

P239boxplot(mattsDF,1,5)        #Create a boxplot of the data in column
                                #1, using the factors in column 5.
```

P239meanbarplot *Produce a series of barplots showing mean values across subjects and output to TeX File* P239meanbarplot

Description

Very similar to P239barplot, except that the subject pool means are plotted and standard error bars are included in the plots.

Usage

```
P239meanbarplot(FileName,DF,ordervect,SEMs,ascending=F)
```

Arguments

`FileName`	A string containing the name of the TeX file to be produced, excluding the extention.
`DF`	The dataframe containing the data to be plotted. This dataframe should contain the subject pool mean values. The data is plotted from columns 4,7,10,13, against the factors in column 14. This will be generalized
`ordervector`	A numeric vector containing the order in which the rows in DF should be plotted. This vector must have the same length as the number of rows in the dataframe and must contain all of the rownumbers.
`SEMs`	This is a dataframe which has the exact same format as DF, except that it contains Standard Error values for the subject pools.
`ascending`	This is a logical value which indicates whether or not the barplot should be reordered in order of ascending value. If TRUE, this overrides the ordervector, but the ordervector must still be passed. This defaults to FALSE.

Value

None. The series of barplots is appended to the file specified.

Examples

```
#DF is a Data frame with 6 insertions, numeric values in column 1
#and factors in column 5.  DF2 is a dataframe with the exact same
#structure, but contatining standard error values for the groups

P239meanbarplot("/mnt/D/temp/test",DF,c(4,5,6,3,2,1),DF2,F)
```

`P239pairwise`	*Pairwise comparisons on subsets of a dataframe*	`P239pairwise`

Description

This function is used to perform multiple (all) pairwise comparisons on a subset of a data frame. The data frame is assumed to be in the form output by stackandbox The subset column is specified in "bycol" and the subset values in "subsetval" (as a vector)the p.adjust.method can be passed via "method" but is by default "bonferroni"

Usage

```
P239pairwise(DF,bycol,subsetval,method="bonf")
```

Arguments

`DF`	The dataframe containing the data to be compared
`bycol`	The number of the column to be used for grouping.
`subsetval`	A vector containing the values (in bycol) which should be included for comparisions. The mode of this vector should be the same as the mode of the data in bycol.
`method`	A string indicating which p-value adjustment is to be used Valid strings are those accepted by pairwise.t.test. Default value is "bonf".

Value

A matrix showing all possible pairwise comparisons with adjusted p-values.

See Also

P239 t.test

Examples

```
#mattsDF is a Data frame with numeric values in column 1 and
#factors in column 5

P239pairwise(mattsDF,1,5,"bonf")
```

P239plot — *Plot Pre and Post-tilt values for any variable, one line for each subject, one plot for each experimental run* — P239plot

Description

This function was used to create a series of plots showing pre and post tilt values of some variable for all P239 runs. For each experimental run, a plot was produced with 8 lines (one for each subject). The line connects two points; the pre value and the post value. In this manner, the trend observed can be compared accross subjects for each run type. The labels along the x axis are taken from the column names of the dataframe. The plots are sent to a Tex file as a table of plots. Currently, the name of the tex file is statically set as FirstTry but this can be generalized later. The user is prompted to input a title for the table of plots.

Usage

```
P239plot(DF,x1,x2,xfacts,ylabel,xlabel)
```

Arguments

`DF`	The dataframe containing the data and factors
`x1`	A numeric indicating the column of DF to use as the first (ie. pre) point
`x2`	A numeric indicating the column of DF to use as the second (ie. post) point.
`xfacts`	A numeric indicating the column of DF to use as the factors (ie. run type). For each unique factor, a plot will be produced showing one line for each subject.
`ylabel`	A string indicating the y axis label
`xlabel`	A string indicating the x axis label

Value

None. A Tex file containing a series of plots is produced.

See Also

P239barplot, P239boxplot,P239meanbarplot

Examples

```
#mattsDF is a dataframe with column 1 and 2 containing numeric values
#(say, systolic blood pressure) and column 3 containing factors
#We'll assume column 1 is the "pre tilt value" and column 2 is the
#"post tilt" value. For each unique level of factors in column three,
#a series of lines will be produced (one for each insertion) connecting
#the pre and post tilt.

P239plot(mattsDF,1,2,3,"Systolic Blood Pressure","Pre or Post Window")
```

`P239subset` *Select a subset of a dataframe by specifying factors* `P239subset`

Description

This function is provided as a convenient method to extract a subset of rows from a dataframe.

Usage

```
P239subset(DF,subsetcol,subsetval)
```

Arguments

`DF`	The original dataframe.
`subsetcol`	The number of the column which is to be used for selecting the subset of rows.
`subsetval`	A vector containing the values of data in subsetcol which will included in the returned dataframe. The mode of this argument must be the same as the mode of data in subsetcol.

Value

A dataframe with the desired subset of rows from the original dataframe.

Examples

```
#mattsDF is a Data frame with numeric values in column 1 and factors
#in column 3. One of the factors in column 3 is "PPB"
PPBonly<-P239subset(mattsDF,3,"PPB")     #PPBonly now contains all rows
                                         #where column 3 contained "PPB"
```

`TSTATgetdata` *Call TSTAT to get some data* `TSTATgetdata`

Description

This is a routine to get data from a datatrac file, using TSTAT Note: the desired insertion must be set before calling this function using the TSTAT function select() It is possible to get data using the more generic callTSTAT function, but since it it frequently necessary to get data, TSTATgetdata was developed to make this specific task easier. Also, this function recognizes the channel names of rr (rr interval), hr (heart rate), pp (pulse pressure), map (mean arterial pressure), and SI (stiffness index) and automatically calls the correct routine to convert the blood pressure data or ECG peak data to the desired data.

Usage

```
TSTATgetdata(TSTATname,ProfName,ChName)
```

Arguments

`TSTATname`	A string containing the name of a valid TSTAT object
`ProfName`	A string containing the name of a valid Profile in the TSTAT object
`ChName`	A String containing the name of the desired data channel. It must be either the proper TAG for the Datatrac file or a custom one which is handled here (currently rr for RR interval, hr for heart rate, pp for pulse pressure) Note, by convention, UPPER CASE letters are used for proper datatrac tags, while lower case are used for channels that must be calculated

Value

An array containing the desired channel data. The first column is the timestamps, and the second is the data values

See Also

callTSTAT,ecgp2hr,ecgp2rr,sysdia2pp,sysdia2map,sysdia2si

Examples

```
#mattsobj is a TSTAT object with the profile window "Whole" defined and
#an insertion named "SomeInsertion"   The datatrac files contain a channel
#Tagged "HYS"

mattsobj$select("SomeInsertion")
someSYSdata<-TSTATgetdata("mattsobj","Whole","HYS")      #returns the HYS data
                                                         #in window Whole for
                                                         #insertion SomeInsertion
```

`TexCurFig`	*Send current figure to a LaTeX file*	`TexCurFig`

Description

TexCurFig is used to send the current active figure to a LaTeX file. the figure is first saved to a postcript file. The only required input is the TeX file name The image is appended to the TeX File if it exists

Usage

`TexCurFig(TexFileName)`

Arguments

`TexFileName`	A string indicating the name of the TeX File, excluding the .tex extension which is included automatically. The figure is save to the file TexFileName# ps where # is the next available filename.

Value

None As a side effect, a postscript file is created and either a new TeX File is created or some text is appended to the end of an existing one.

Examples

`TexEndTableFig` *End a table of figures in a TeX file.* `TexEndTableFig`

Description

TexEndTableFig is used to end a table of figures in a TeX File which was started with TexStartTableFig.

Usage

`TexEndTableFig(TexFileName)`

Arguments

`TexFileName`	A string indicating the name of the TeX File, excluding the .tex extension which is included automatically.

Value

None Some text is added to an existing TeX File

Examples

`TexFindPSNum` *Find the next available postscript file number for a given file name* `TexFindPSNum`

Description

TexFindPSNum is used to find the next available postscipt file number for a given file name

Usage

`TexFindPSNum(TexFileName)`

Arguments

`TexFileName`	A string indicating the name of the TeX File, excluding the tex extension which is included automatically. The next available file name of the type TexFileName# ps is returned as a result.

Value

The next available file name for a postscript file.

Examples

`TexRMAOV` *Send an Analysis of Variance Summary with more than one error stratum to a LaTeX file* `TexRMAOV`

Description

TexRMAOV is used to summarize a Repeated Measures Analysis of Variance object and send the output to a LaTeX file. The only required input is the TeX file name. Alternatively, any type of AOV with more than one error stratum can be sent.

Usage

```
TexRMAOV(TexFileName, AOVobject)
```

Arguments

`TexFileName`	A string indicating the name of the TeX File, excluding the .tex extension which is included automatically
`AOVobject`	An object of class AOV.

Value

None As a side effect either a new TeX File is created or some text is appended to the end of an existing one

Examples

`TexSimpleAOV` *Send an Analysis of Variance Summary with one error stratum to a LaTeX file* `TexSimpleAOV`

Description

TexSimpleAOV is used to summarize a one stratum Analysis of Variance object and send the output to a LaTeX file. The only required input is the TeX file name

Usage

```
TexSimple(TexFileName, AOVobject)
```

Arguments

`TexFileName`	A string indicating the name of the TeX File, excluding the .tex extension which is included automatically.
`AOVobject`	An object of class AOV.

Value

None. As a side effect either a new TeX File is created or some text is appended to the end of an existing one.

Examples

`TexStartTableFig` *Begin a table of figures in a TeX file.* `TexStartTableFig`

Description

TexStartTableFig is used to begin a table of figures in a TeX File

Usage

```
TexStartTableFig(TexFileName, numCols, TableTitle)
```

Arguments

`TexFileName`	A string indicating the name of the TeX File excluding the .tex extension which is included automatically.
`numCols`	A numeric indicating the numbner of coulumns the table should have
`TableTitle`	A string indicating the desired Title for the table.

Value

None. Either a new TeX File is produced or some text is added to an existing one.

Examples

`TexSummaryAOV` *Send an Analysis of Variance Summary to a LaTeX file* `TexSummaryAOV`

Description

TexSummaryAOV is used to summarize an Analysis of Variance object and send the output to a LaTeX file. The only required input is the TeX file name. Depending on the number of error strata, either TeXSimplAOV of TexRMAOV is called. Either of these could be called directly. Refer to the appropriate section

Usage

```
TexSummaryAOV(TexFileName, AOVobject)
```

Arguments

`TexFileName`	A string indicating the name of the TeX File, excluding the .tex extension which is included automatically.
`AOVobject`	An object of class AOV.

Value

None. As a side effect either a new TeX File is created or some text is appended to the end of an existing one.

Examples

`TexTable`	*Send a table or matrix to a TeX file*	`TexTable`

Description

TexTable is used to send a table of data to a LaTeX file.

Usage

```
TexTable(TexFileName, VectorData, Numcols)
```

Arguments

`TexFileName`	A string indicating the name of the TeX File, excluding the .tex extension which is included automatically.
`VectorData`	An data frame or array of numerics.
`Numcols`	A numeric indicating the number of coums to take from VectorData

Value

None. As a side effect either a new TeX File is created or some text is appended to the end of an existing one

Examples

`TexTableFig` *Add the current figure to a table of figures in a TeX file.* `TexTableFig`

Description

TexTableFig is used to add a figure to a table of figures in a TeX File which was started with TexStartTableFig

Usage

```
TexTableFig(TexFileName, caption)
```

Arguments

`TexFileName`	A string indicating the name of the TeX File, excluding the .tex extension which is included automatically.
`TexFileName`	A string indicating the caption that should be used for the figure.

Value

None. Some text is added to an existing TeX File and a new Postscript file is produced for the current figure.

Examples

`Texify` *Compile a LaTeX file and send the output to the default DVI viewer.* `Texify`

Description

Texify is used to process a raw TeX file and send the output the the default DVI viewer LaTeX file.

Usage

```
Texify(TexFileName)
```

Arguments

`TexFileName`	A string indicating the name of the TeX File, excluding the .tex extension which is included automatically.

Value

None As a side effect a new DVI file is created and displayed.

Examples

`Vdecimate` *Decimate a vector of data* `Vdecimate`

Description

This function is used to decimate a vector of data

Usage

```
Vdecimate(Vectorname,fact=10)
```

Arguments

`Vectorname`	A vector of data, NOT a character string
`fact`	A numeric specifying the decimation factor Defaults to 10.

Value

A numeric vector containing the decimated data

Examples

```
Vdecimate(c(1,2,3,4,5,6,7,8,9),2) #would return 1,3,5,7,9
```

`Vintegratefull` *Integrate a vector of data above or below a reference value* `Vintegratefull`

Description

This function is used to perform an integration along a vector of data. The "full" refers to the fact that it can integrate below or above a given value (not both,ie. negative integrals don't cancel positive)

Usage

```
Vintegratefull(Vname,Vtimes,ExtraVect,direction, delta=1)
```

Arguments

`Vname`	A numeric vector (Not the name of the vector) containing the curve to integrate above or below.
`Vtimes`	The timestamps corresponding to the vector Vname
`Extravect`	A numeric vector of length 3 The elements are start time of integration, stop time of integration, and reference value to integrate above or below, in that order.
`direction`	A character string indicating the direction to integrate. Acceptable values are "below" or "Below" otherwise above is assumed.
`delta`	A numeric indicating the factor which should be used to sale the area Mainly usefull if Vtimes is simply an index vector and the spacing is not one unit (eg 1 second).

Value

A numeric representing the area of the integral. A plot is produced to display the area calculated.

Examples

```
#BPvalues is a vector of blood pressures and BPtimes is a vector containing
#the corresponding timestamps

Vintegratefull(BPvalues,BPtimes,c(50,60,100),"below")
#Integrate the blood pressure waveform between 50 and 60 seconds, below
#100 mmHg
```

`addfact` *Function to loop through a data frame and prompt for factor levels* `addfact`

Description

Addfact was written to make it easier to add a column of factors to a data frame. The function is passed a dataframe and a vector of strings which are to be used as factors The function then prompts for each insertion, to select a factor. Note the original dataframe is not altered. Rather, the result must be cbinded to the dataframe manually.

Usage

```
addfact(DFname,lnames)
```

Arguments

`DFname`	A data frame, not just its name
`lnames`	A vector of strings which contains all of the deired levels of the factor

Value

A column of factors, the same length as the number of rows in the dataframe

See Also

addnumeric

Examples

```
#mattsDF is a Data frame with numeric values in column 1.  Add a column
#of factors with the levels "Relax","Control","PPB","Gsuit",and "PPB&GS"

ExpType<-addfact(mattsDF,c("Relax","Control","PPB","Gsuit","PPB&GS"))
mattsDF<-cbind(mattsDF,ExpType)
```

PS17308 PDF [Page: 161 of 248]

`addnumeric` *Function to loop through a data frame and prompt for numeric levels* `addnumeric`

Description

Similar to addfact, addnumeric was written to make it easier to add a column of numbers to a data frame. The function is passed a dataframe and a vector of numerics which are to be used as the levels of the numbers The function then prompts for each insertion, to select a level. This can be useful to add a column of data which are to be used as regressors. Note the original dataframe is not altered. Rather, the result must be cbinded to the dataframe manually.

Usage

```
addnumeric(DFname,lnames)
```

Arguments

`DFname`	The actual dataframe, not just its *name*
`lnames`	A vector of numerics which contains all desired levels of the column.

Value

A column of numerics, the same length as the number of rows in the dataframe.

See Also

addfact

Examples

```
#mattsDF is a Data frame with numeric values in column 1  Add a column
#of numerics with the levels 0,4,5

GSlevel<-addnumeric(mattsDF,c(0,4,5))
mattsDF<-cbind(mattsDF,GSlevel)
```

`calcslope` *Calculate the best fit slope of the line between two vectors* `calcslope`

Description

This is a simple function to calculate the slope of the line between two vectors of points. The function calls R's lm function.

Usage

```
calcslope(y,x)
```

Arguments

`y`	A numeric vector representing the dependant variable
`x`	A numeric vector representing the dependant variable

Value

A numeric representing the slope,m, of y=mx

See Also

lm

Examples

```
calcslope(c(20,40,60,80),c(1,2,3,4))     #would return a linear model with slope
                                          #of 20 and intercept of zero.
```

`callTSAT` *Wrapper function to enable executing TSTAT commands in loops* `callTSAT`

Description

This is a wrapper function to call any TSTAT function for the specified TSTAT object It's main purpose is to allow TSTAT function calls to be specified and made in loops and the required calls to be determined programaticallyt

Usage

```
callTSTAT(ObjectName,FunName,Arg1="",Arg2="",Arg3="",...)
```

Arguments

`Objectname`	A string containing the name of a valid TSTAT object
`FunName`	A string containing the name of a valid TSTAT function to be called via ObjectName$FunName
`Arg1, Arg2, ...`	Strings containing arguments required for the TSTAT function Refer to TSTAT manual for details on specific functions. Although there is potentially no limit to the number of arguments, currently, this function returns an error if more than five are passed. This could be changed in the future

Value

Whatever value is returned by the chosen TSTAT function. Refer to TSTAT manual for more details.

See Also

TSTATgetdata

Examples

```
#mattsobj is a TSTAT object with the profile window "Whole"
#defined and an insertion named "SomeInsertion".  The datatrac
#files contain a channel Tagged "HYS"

callTSTAT("mattsobj","select","SomeInsertion")
#use call TSTAT to select the current insertion

#compare the above method to
mattsobj$select("SomeInsertion")
#Although the callTSAT version appears more complicated, it
#can be directly called in loops with different strings
#for the insertion names

data<-callTSTAT("mattsobj","getd","Whole","HYS")
#returns the HYS data in the "Whole"
dow, for the current insertion

TSTATinfo<-callTSTAT(TSTATname,"get");
#returns the entire set of information about a TSTAT
#object as a list structure
```

---

dynamicpts *Syncronize and Interpolate Insertions* dynamicpts

---

Description

This function syncronizes data from each insertion and interpolates the data to each second. The function accesses the "Whole" profile of TSTAT, and uses the start times and Timepts to determine which insertion has the least amount of data before the tilt. Similarly, the least amount of data after tilt is determined. All insertions are processed using these time limits. This does discard data from some insertions, but avoids the problems associated with NA's later on.

Usage

`dynamicpts(TSTATobject,Timepts,ChName,plot=TRUE)`

Arguments

`TSTATobject`	A string which contains the name of a TSTAT object
`Timepts`	Timepts is a numeric vector or matrix.The first column of Timepts must contain the point to syncronize to (usually start of Tilt) for each insertion Any other columns are ignored.
`ChName`	A string which contains the Datatrac TAG or a custom name which can be handled by TSTATgetdata(), eg. rr.hr.pp, etc
`plot`	if TRUE, will produce a plot of the delta runs

Value

A Dataframe with a special structure. Second-by-second data appears in columns (ie column one is second 1, column 2 is second 2. etc. The first row contains the timesatmps, and the second row contains the data for the reference channel. By default, the function uses the GZ channel as the reference channel. This could be generalized later.

See Also

dynplot. dynplotdelta

Examples

```
#mattsobj is a TSTAT object with 6 insertions and a profile titled
#"Whole" which has been defined to contain the entire signal.
#GZtimes is an array containing the times of GZ transitions for each
#insertion

DynamicBP<-dynamicpts("mattsobj",GZtimes[,1:2],"HYS")  #Calculate the second-
                                                        #by-second systolic blood pres-
sure and
                                                        #syncronize insertions using the start to
                                                        #head-up to head-down tilt
```

`dynplot`	*Plot dynamic (second by second) data*	`dynplot`

Description

The routine uses the rownames of the input dataframe to group insertions and calculate pooled data. The group pool mean and standard deviation are calculated at each second and ploted for the cases in showcases. The input dataframe is typically generated by the function dynamicpts. However, since this function utilizes the rownames of the input dataframe, it is important to ensure that they are set correctly to identify the group names.

Usage

```
dynplot(dyndata,showcases="all",maintitle="",xl="",yl="",ylim=c(20,200))
```

Arguments

`dyndata`	A dataframe containing second by second dynamic data Refer to function dynamicpts for the structure of the dataframe
`showcases`	An (optional) vector of strings which indicates which groups should be plotted. The groups are determined from the rownames of dyndata If not entered, all groups will be plotted, possible resulting in a cluttered plot
`maintitle`	The title of the plot.
`xl`	An (optional) character string which states the label for the x-axis
`yl`	An (optional) character string which states the label for the y-axis.
`ylim`	An (optional) vector indicating the dimensions of the y-axis. By default, it is 20 to 200.

Value

None A plot is produced. diplaying the dynamic data for all cases in showcases. The function plots the pool mean +/- standarad error for groups which are determined by the rownames of dyndata.

See Also

dynamicpts, dynplotdelta

Examples

```
#DynamicBP is a dataframe generated by dynamicpts.  Refer to that function
#for details on the structure.  The rownames must be set to identify the
#experiment type (or other relevent grouping).
rownames(DynamicBP)<-c("PPBduringHDT","NoPPB", ..)      #Replace array elements with
                                                        #applicable names
dynplot(DynamicBP,c("PPBduringHDT"))            #to see only PPBduringHDT runs
dynplot(DynamicBP,c("NoPPB"))                   #to see only NoPPB runs
dynplot(DynamicBP,c("PPBduringHDT","NoPPB"))    #to see both runs on the same plot
```

`dynplotdelta` *Calculate dynamic (second by second) delta values and plot* `dynplotdelta`

Description

This routine is essentially the same as dynplot, except that it first calculates the delta values for the second by second data. The reference rest level is calculated from the first "restlength" data points for each insertion.

Usage

```
dynplotdelta(dyndata,restlength,showcases="all",maintitle="",xl="",yl="")
```

Arguments

`dyndata`	A dataframe containing second by second dynamic data. Refer to function dyndata for the structure of the dataframe.
`restlength`	The number of points from the begining of the dynamic data to use for the reference level. The mean of these first "restlength" points is used as the reference.
`showcases`	An (optional) vector of strings which indicates which groups should be plotted. As in dynplot, the groups are determined from the rownames of dyndata. If not entered. all groups will be plotted, possible resulting in a cluttered plot.
`maintitle`	The title of the plot
`xl`	An (optional) character string which states the label for the x-axis.
`yl`	An (optional) character string which states the label for the y-axis.

Value

A dataframe of the same size and structure of dyndata, containing second by second delta values which are the difference between the reference value and the dynamic data values.A plot is produced, diplaying the dynamic delta values for all cases in showcases. The function plots the pool mean delta +/- standarad error for groups which are determined by the rownames of dyndata

See Also

dynamicpts, dynplot

Examples

```
#DynamicBP is a dataframe generated by dynamicpts.  Refer to that function
#for details on the structure.  The rownames must be set to identify the
#experiment type (or other relevent grouping)
rownames(DynamicBP)<-c("PPBduringHDT","NoPPB", ..)      #Replace array elements with
                                                         #applicable names

DeltaBP<-dynplotdelta(DynamicBP,4,c("PPBduringHDT","NoPPB"))

#Note, that the 4 is the length of the resting period used for baseline (calculated at the
#beginning of the data sets).  Also note, that the y-axis of the graph has been statically
#set to a poor scale (which was used for ECG signal), so the graph looks awful.
#This will be fixed.
```

---

ecgp2hr *Convert ECG peak values to heart rate* ecgp2hr

Description

As the name suggests, this function takes a matrix of ECGP (ECG peaks) data obtained by TSTAT and returns a vector of heart rates with timestamps. The ECGP matrix must be organized as in the datatrac files, ie first column=timestamps, second column=numeric level of peak

Usage

`ecgp2hr(ECGPmatrix)`

Arguments

`ECGPmatrix`	The ECGP matrix as found in the dattrac file. See format above

Value

A matrix of heart rates with the same structure as ECGP

See Also

ecgp2rr

Examples

```
#mattsobj is a TSTAT object with 6 insertions and a window titled "Whole" which has been de-
fined to contain the entire signal.
ECG<-TSTATgetdata("mattsobj","Whole","ECG")      #First get the data
ECGP<-TSTATgetdata("mattsobj","Whole","ECGP")    #The ECG peaks were calculated before-
hand and                                                                   #saved in the data-
trac file

plot(ECG,type="l",ylim=c(-1,1),xlim=c(0,70))     #plot the data
par(new=T)
plot(ECGP,type="p",ylim=c(-1,1),xlim=c(0,70),col="red")

HR<-ecgp2hr(ECGP)                                #calculate Heart Rate
par(new=T)
plot(HR,type="l",ylim=c(0,2),xlim=c(0,70),col="blue")
```

---

ecgp2rr *Convert ECG peak values to RR intervals* ecgp2rr

Description

As the name suggests, this function takes a matrix of ECGP (ECG peaks) data obtained by TSTAT and returns a vector of RR intervals with timestamps. The ECGP matrix must be organized as in the datatrac files, ie. first column=timestamps, second column=numeric level of peak.

Usage

`ecgp2rr(ECGPmatrix)`

Arguments

`ECGPmatrix`	The ECGP matrix as found in the dattrac file. See format above.

Value

A matrix of RR intervals with the same structure as ECGP

Examples

```
ECG<-TSTATgetdata("mattsobj","Whole","ECG")      #First get the data
ECGP<-TSTATgetdata("mattsobj","Whole","ECGP")    #The ECG peaks were calculated before-
hand and                                                                   #saved in the data-
trac file.

plot(ECG,type="l",ylim=c(-1,1),xlim=c(0,70))     #plot the data
par(new=T)
plot(ECGP,type="p",ylim=c(-1,1),xlim=c(0,70),col="red")

RR<-ecgp2rr(ECGP)                                #calculate RR interval
```

```
par(new=T)
plot(RR,type="l",ylim=c(0,2),xlim=c(0,70),col="green")
```

`firstquart` *Calculate the first quartile of a vector of data* `firstquart`

Description

A simple function to return the first quartile of a vector of data. Something of this nature should be considered when trying to extract an index of a signal for statistical analysis, rather than the minimum which is prone to detecting outliers. This function calls R's more general quantile function.

Usage

```
firstquart(x)
```

Arguments

`x`	A numeric vector for which the first quartile is to be calculated

Value

The first quartile of the input vector, x.

Examples

```
firstquart(c(0,10,20,30,40,50,60,70,80,90,100)) #would return 25
```

`firsttry` *A wrapper function for Dfapplyfun* `firsttry`

Description

firsttry is yet a higher level interface to Dfapplyfun. The purpose is to calculate some index from the data in a given window which can be specified as a string (eg "pre", "post", "hdt", or "pre&hdt"). The window times are then calculated based in the input GZtimes Documentation for this will be completed later.

Usage

```
firsttry(TSTATobject,Channel,GZtimes,win,windowlength,startnum=1,endnum=207,applyfunction)
```

Arguments

`TSTATobject`	A string naming the TSTAT object
`Channel`	A string containing the TAG for the channel in the datatrac file
`GZtimes`	An array of transition times with one row for each insertion. These are normally calculated using Dfapplyfun and Gzpoints Currently the routine handles only the type of tilt-table profiles conducted for protocol 239; head-up to head-down to head-up. Refer to Dfapplyfun and Gzpoints for more details
`win`	A string naming the desired window Valid strings are "pre","post"."hdt","pre&hdt"
`windowlength`	A numeric indicating the window length to be used. This is necessary for all but "hdt" win's. For a "pre" tilt the data used will start windowlength seconds before the HU-HD transition and end at the HU-HD transition. For a "post" tilt the data used will start at the HD-HU transition and end windowlength seconds after. For a "pre&hdt" tilt the data used will begin windowlength seconds before the HU-HD transition and end windowlength seconds after the HD-HU transition. This function is not really intended to be used in the final package as it is just an extension of Dfapplyfun
`startnum`	The first insertion number to process.
`endnum`	The last insertion number to process. The default is 207 because there were 207 insertions in the P239 dataset.
`applyfun`	A string naming a function that Dfapplyfun can handle such as min, max,etc.

Value

A dataframe containing the results calculated for each insertion

See Also

Dfapplyfun, refirsttry

Examples

```
#mattsobj is a TSTAT object with the "Whole" profile window defined.
#GZtimes is a (#of insertions)x4 array containing the GZ transition times
firsttry("mattsobj","HYS",GZtimes,"pre",5,1,207,"min")  #cycles through insertions
#calculating the minimum systolic BP (HYS) in the 5 second window
#preceding the HU-HD transition

firsttry("mattsobj","HYS",GZtimes,"post",15,1,207,"Vintegratefull")
#cycles through insertions calculating the maximum systolic BP (HYS)
#in the 15 second window following the HD-HU transition
```

groupdata *Group and rename data from different groups in a dataframe* groupdata

Description

This function is used to organize various levels of factors into one of two groups for comparison

Usage

```
groupdata(DF,bycol,Group1,Group2,Groupnames)
```

Arguments

`DF`	The original data frame containing data and factors. The first two rows are assumed to be Timestamps and GZ and are ignored. This could be generalized later.
`bycol`	A numeric specifying which columd of DF contains the factors for grouping
`Group1`	A vector of strings. If the factor of particular entry in DF matches one of the strings, it will be classified as Group1
`Group2`	A vector of strings If the factor of particular entry in DF matches one of the strings, it will be classified as Group2
`Groupnames`	A vector of two strings which will be used to name the two groups

Value

A copy of the original dataframe with a factor column cbinded. the factor column contains the groupnames

See Also

P239subset

Examples

```
#mattsobj is a datframe with the third column containing factors  The factors contain
#various combinations of PPB and Gsuit.  This function could classify the insertions into
#either PPB or Gsuit.
groupdata(mattsobj,3,c("PPBduringHDT","PPBafterHDT"),c("GS5psi","GS4psi"),c("PPB","Gsuit"))
```

`last` *Return the last value in a vector* `last`

Description

This is a very simple program to return the last value in an array.

Usage

```
last(x)
```

Arguments

`a`	A vector of any type.

Value

The last value in the vector.

Examples

```
last(c(1,3,4,2))  #would return 2
```

`multTest` *Multivariate paired T^2-test* `multTest`

Description

This function performs a multivariate equivalent of a paired t-test on two dependent groups of data from a data frame. This test is similar to the "P239 t.test" command

Usage

```
multTest{DF,datacol,bycol,subsetval}
```

Arguments

`DF`	The dataframe containing the data and the groups (factors)
`datacol`	A numeric vector indicating the column numbers of DF which contains the data.
`bycol`	A numeric indicating the column number of DF which contains the group names (factors)
`subsetval`	A vector containing two strings, representing the names of groups to compare.

Value

An object consisting of the p-value and the test statistic and the confidence interval (all numeric)

`refirsttry` *Almost identical to firsttry, this function repeats the work done by firsttry. However, this function allows for rejecting points in the signals before calculating values* `refirsttry`

Description

Very similar to firsttry, except that this function is intended to allow the user to revisit one or more insertions and exclude some outlier points before the index is calculated. Refer the firsttry for a detailed explanation of usage and examples Again, this function is not really intended to be used in the final package as it is just an extension of Dfapplyfun.

Usage

Arguments

Value

See Also

Dfapplyfun, firsttry

Examples

`stackandbox` *Stack dynamic (second by second) data into a long dataframe and produce boxplot* `stackandbox`

Description

This function is used to rearrange a data frame of dynamic points into a form suitable for boxplotting and/or analyses of variance The first two rows are ignored since they are assumed to be timestamps and GZ data .Each column is stacked below the previous to form a long data frame to which a vector of Factors can be cbinded. The rownames are repeated as the rownames of the reulting data frame. At the end the result is boxplotted against the rownames, and the stacked data frame returned

Usage

```
stackandbox(DataFrame,subnames="")
```

Arguments

`DataFrame`	The original dataframe.
`subnames`	Currently not used. Will be used later to select a subset of rows from the original dataframe.

Value

A tall dataframe with each column from the original dataframe stacked vertically. Also, a boxplot is produced.

See Also

dynamicpts, dynplot

Examples

```
#dataFrame is a dataframe which has been generated by the function dynamicpts
stackedData<-stackandbox(DataFrame)  #produces a tall dataframe and a boxplot
```

`sysdia2map` *Convert systolic and diastolic data to mean arterial pressure* `sysdia2map`

Description

As the name suggests, this function takes a vector of SYS (Systolic Blood Pressure) and DIA (Diastolic Blood Pressure) data obtained by TSTAT and returns a vector of mean arterial Pressures with timestamps.

Usage

```
sysdia2map(SYS,DIA)
```

Arguments

`SYS`	A matrix of Systolic blood pressures as obtained by TSTAT, ie. timestamps, Blood Pressure
`DIA`	A matrix of Diastolic blood pressures as obtained by TSTAT, ie. timestamps, Blood Pressure

Value

A matrix of mean arterial pressures with the same structure as SYS.

See Also

sysdia2pp, sysdia2si

Examples

```
SYS<-TSTATgetdata("mattsobj","Whole","SYS")            #first get data
DIA<-TSTATgetdata("mattsobj","Whole","DIA")            #The systolic and diastolic were calculate
                                                       #beforehand and saved in the data-

trac file.

plot(SYS,type="l",ylim=c(50,130),xlim=c(0,70))  #Plot data
par(new=T)
```

```
plot(ECGP,type="p",ylim=c(50,130),xlim=c(0,70),col="blue")

#Now, calculate the mean arterial pressure:

MAP<-sysdia2map(SYS,DIA)                                #calculate mean arterial pressure
par(new=T)
plot(MAP,type="l",ylim=c(50,130),xlim=c(0,70),col="green")
```

`sysdia2pp` *Convert systolic and diastolic data to pulse pressure* `sysdia2pp`

Description

As the name suggests, this function takes a vector of SYS (Systolic Blood Pressure) and DIA (Diastolic Blood Pressure) data obtained by TSTAT and returns a vector of Pulse Pressures with timestamps.

Usage

```
sysdia2pp(SYS,DIA)
```

Arguments

`SYS`	A matrix of Systolic blood pressures as obtained by TSTAT, ie. timestamps. Blood Pressure
`DIA`	A matrix of Diastolic blood pressures as obtained by TSTAT, ie timestamps, Blood Pressure

Value

A matrix of pulse pressures with the same structure as SYS

See Also

sysdia2map, sysdia2si

Examples

---

sysdia2si *Convert systolic and diastolic data to stiffness index* **sysdia2si**

Description

As the name suggests, this function takes a vector of SYS (Systolic Blood Pressure) and DIA (Diastolic Blood Pressure) data obtained by TSTAT and returns a vector of Pulse Pressures with timestamps.

Usage

```
sysdia2si(SYS,DIA)
```

Arguments

`SYS`	A matrix of Systolic blood pressures as obtained by TSTAT, ie. timestamps, Blood Pressure
`DIA`	A matrix of Diastolic blood pressures as obtained by TSTAT, ie. timestamps, Blood Pressure

Value

A matrix of Stifness Index with the same structure as SYS

See Also

sysdia2map, sysdia2pp

Examples

```
SYS<-TSTATgetdata("mattsobj","Whole","SYS")              #first get data
DIA<-TSTATgetdata("mattsobj","Whole","DIA")              #The systolic and diastolic were calculated
                                                         #beforehand and saved in the data-
trac file.

plot(SYS,type="l",ylim=c(50,130),xlim=c(0,70))           #Plot data
par(new=T)
plot(ECGP,type="p",ylim=c(50,130),xlim=c(0,70),col="blue")

#Now, calculate the Stiffness Index:

SI<-sysdia2si(SYS,DIA)
par(new=T)
plot(SI,type="l",ylim=c(200,700),xlim=c(0,70),col="purple")
```

`verifyDFC` *Graphically verify DFC data for a TSTAT object* `verifyDFC`

Description

This function is used to verify the DFC data which may have been obtained directly by TSTAT. However, TSTAT doesn't offer visual verification of the value obtained, so this function has been provided. This function only plots the results for one insertion so, in general, it will be called in a loop to cyle through all insertions. For the current insertion, this function plots the entire channel data in blue and the data for the profile window is overlapped in red. The value calculated by TSTAT is shown as a horizontal red line.

Usage

```
verifyDFC(DFCdata,Profiledata,Wholedata,starttime)
```

Arguments

`DFCdata`	One numeric data value from the TSTAT command $getDFC
`Profiledata`	A vector of numerics containing the data from the TSTAT profile window for which the DFC data is calculated
`Wholedata`	The entire data signal of the appropriate channel.
`starttime`	A numeric indicating the start time of the TSTAT profile window

Value

None. For the current insertion, the entire channel data, profile data, and calculated DFC value is plotted.

See Also

Dfapplyfun

Examples

```
#mattsobj is a TSTAT object with the profile windows "Whole" and "WindowA" defined
#the TAG for systolic blood pressure (interpolated to 100 Hz) is HYS.  Also, A data frame
#column named "Minimum Sys BP" has been defined in mattsobj.
callTSTAT("mattsobj","select", "S5e5")   #where "S5e5" can be replaced with the
                                        #desired insertion (eg. "S5e1", "S5e11", etc )

wholedata<-callTSTAT("mattsobj","getd","Whole","HYS");
profiledata<-callTSTAT("mattsobj","getd","WindowA","HYS");
DFCdata<-callTSTAT("mattsobj","getDFC","WindowA","Minimum Sys BP");
verifyDFC(DFCdata[[1]][i],profiledata,wholedata,starttime),
#where i corresponds to "S5e5" and starttime is the starting time
#of WindowA which can be found using the $get() functionof TSTAT
#Notice that this process to visually verify ONE DFC value for ONE
#insertion is very involved.
```

Index

addfact, 17, 18
addnumeric, 17, 18

calcslope, 18
callTSAT, 19
callTSTAT, 10
CreatePaper, 1

Dfapplyfun, 1, 26, 29, 33
dynamicpts, 20, 22, 23, 30
dynplot, 21, 23, 30
dynplotdelta, 21, 22

ecgp2hr, 10, 23
ecgp2rr, 10, 23, 24

firstquart, 25
firsttry, 25, 29

groupdata, 27
Gzpoints, 3

last, 28
lm, 19

multTest, 28

P239.t.test, 4, 7
P239barplot, 4, 8
P239boxplot, 5, 8
P239meanbarplot, 6, 8
P239pairwise, 7
P239plot, 8
P239subset, 9, 27

refirsttry, 26, 29

stackandbox, 29
sysdia2map, 10, 30–32
sysdia2pp, 10, 30–32
sysdia2si, 10, 30–32

TexCurFig, 10
TexEndTableFig, 11
TexFindPSNum, 11
Texify, 15
TexRMAOV, 12
TexSimpleAOV, 12
TexStartTableFig, 13
TexSummaryAOV, 13
TexTable, 14
TexTableFig, 15
TSTATgetdata, 9, 19

Vdecimate, 16
verifyDFC, 33
Vintegratefull, 16

```
ESIstat Tutorial 1: Preprocessing, Analysis, Data Frame Operations
====================================================================


#First make sure the libraries are loaded
  library(tstat,logical.return=T)
  library(ESIstat,logical.return=T)


#Next create the TSTAT object and add some insertions
  mattsobj<-icreate("ExampleObject","Whole WA WB",100,"GZ")
  mattsobj$add("S5e1","/mnt/D/p239/Temp/s5_1.jar")
  mattsobj$add("S5e2","/mnt/D/p239/Temp/s5_2.jar")
  mattsobj$add("S5e5","/mnt/D/p239/Temp/s5_5.jar")
  mattsobj$add("S5e10","/mnt/D/p239/Temp/s5_10.jar")
  mattsobj$add("S5e11","/mnt/D/p239/Temp/s5_11.jar")
  mattsobj$add("S5e17","/mnt/D/p239/Temp/s5_17.jar")


#Next identify the profiles that TSTAT uses to select data.
#for each of the ident commands select (from R command line) 3 then 2
#then select the entire data series
#then select (from R command line) 0 then 0
  mattsobj$ident("S5e1")
  mattsobj$ident("S5e2")
  mattsobj$ident("S5e5")
  mattsobj$ident("S5e10")
  mattsobj$ident("S5e11")
  mattsobj$ident("S5e17")


#We have now entered 5 insertions and defined a window called "Whole" for
```

```
#each one.  Next we will use the ESIstat functions to perform an analysis.

#Dfapplyfun
#==========
#First we will let ESIstat find the appropriate GZ transition points.  Note
#that the name of the TSTAT object must be in quotes.  The argument 1 is
#used to decimate the GZ signal.The result should be saved to a variable fo
#ruse later.

  GZtimes<-Dfapplyfun("Gzpoints",1,"mattsobj","Whole","GZ",1,6)

#We need to rearrange GZtimes a little (this will be automated
#later).  Also, a minor bug results in the times being incorrectly
#decimated.  This will be fixed later, but for now, correct the values by
#dividing by 10:
  dim(GZtimes)<-c(4,6)  #make a matrix
  GZtimes<-t(GZtimes)   #transpose
  GZtimes<-GZtimes/10

#Notice that for S1e1 and S1e5 there was no tilt, so no transitions
#were found.  We have to manually enter some times or select times based on
#another reference signal. To select times based on other signals,
#rerun the last command using "GZpoints2" as the function, instead of
#"GZpoints".  This will prompt you to select another reference signal for
#any insertions where a transition wasn't found.  The previous result,
#GZtimes, must be passed via ExtraVect.  Although all insertions will be
#displayed, only the ones that contain NA in GZpoints will be recalculated
```

```
  GZtimes<-Dfapplyfun("Gzpoints2",GZtimes,"mattsobj","Whole","GZ",1,6)

#Again rearrange GZtimes a little. Again, a minor bug results in the times
#being incorrectly decimated.  This will be fixed later, but for now,
#correct the values by dividing by 100:
  dim(GZtimes)<-c(4,6)  #make a matrix
  GZtimes<-t(GZtimes)   #transpose
  GZtimes[1,]<-GZtimes[1,]/100
  GZtimes[3,]<-GZtimes[3,]/100

#Now GZtimes can be used to perform calculations on various windows of the
#insertions.  Suppose you wish to calculate the mean blood pressure in the
#5 seconds pre tilt, for all insertions.  This is accomplished with:

meanBPpre<-Dfapplyfun("calcDFC",GZtimes[,1]-5,"mattsobj","Whole","HYS",1,6,
"mean")

#Normally, there is a pause after each insertion is plotted by this was
#accidentally left out for the initial package.

#Suppose one of the points in meanBPpre was calculated incorrectly due to
#some extreme outlier.  We could recalculate the insertions using a similar
#command as above, substituting the function "recalDFC" for
#"calDFC". However, although all of the functionality is implemented, it
#was disabled during debugging and has not been re-enabled.  Therefore,
#this function will not work as expected yet.

#There are two more function that can currently be called using
```

```
#Dfapplyfun.  They are Vintegratefull and verifyDFC.  Vintegratefull was
#being developed to calculate a new index for the recovery time from a
#push-pull transition, namely the integral under the resting blood pressure
#after the transition.  For a rough example try:

  integralBPpre<-Dfapplyfun("Vintegratefull", cbind(GZtimes[,3],
  GZtimes[,3]+10,100),"mattsobj","Whole","HYS",1,6,"below")

#The main differences here are that the third argument has an extra column
#which is the number to integrate above or below, and the final argument is
#the direction.  Suitable values are "below" or "Below", otherwise above is
#assumed.

#Finally, the Dfapplyfun can implement verifyDFC for each insertion. This
#function is used to visually verify a data frame column that TSTAT
#calculates.  For example, we could use TSTAT to calculate the minimum
#blood pressure for each insertion using:

  mattsobj$addDFC("MinBP","HYS","min")

#You can see the data calculated if you print mattsobj. Alternatively, you
#could extract the DFC data for the window of interest to a temporary
#variable:

  temp<-mattsobj$getDFC("Whole","MinBP")

#The problem is that you cannot visualize the data using TSTAT.  ESIstat
#utilizes TSTAT to visualize the data:
  Dfapplyfun("verifyDFC",NULL,"mattsobj","Whole","HYS",1,6,"MinBP")

#Note that the calculated value is shown as a red horizontal
#line in the bottom graph.


#Data Frame operations
#======================
#Next, we need to perform some statistical analysis on the data. First, we
#we must create a data frame containing meanBPpre.  This involves creating
#a data frame and adding columns of factors. There are several ways to do
#this, but one is shown here:

  mattsDF<-as.data.frame(meanBPpre)
  rownames(mattsDF)<-c("S5e1","S5e2","S5e5","S5e10","S5e11","S5e17")

#One way to add a column of factors is to simply cbind a vector of
#strings to it:

  mattsDF<-cbind(mattsDF,c("Relax1","Control1-1","TGS-C0","TPP-A4-1",
  "TPP-A4-2","TPGA77-2"))

  names(mattsDF)<-c("meanBPpre","ExpCode")

#for data frames with many insertions, it can be tedious to create the
#vector of factors as shown above.  For convenience, a function is included
#in ESIstat to prompt for each factor from a given list of possible ones:
  ExpType<-addfact(mattsDF,c("Relax","Control","PPB","Gsuit","PPB&GS"))
  mattsDF<-cbind(mattsDF,ExpType)

#Similarly, to add a vector of numeric factors:

  GSlevel<-addnumeric(mattsDF,c(0,4,5))
  mattsDF<-cbind(mattsDF,GSlevel)

#It can be awkward to select rows of a data frame which have one level of a
#certain factor.  ESIstat provides one simple method.  To select rows of
#mattsobj which are of type "PPB" use:

  PPBonly<-P239subset(mattsDF,3,"PPB")

#To perform a simple T-test on a dataframe, ESIstat provides a simple
#method.  However, for our example, there are not two factors which appear
#more than once.  Let's add a fictitious factor:

  mattsDF<-cbind(mattsDF,c("fact1","fact1","fact1","fact2","fact2","fact2"))
  names(mattsDF)<-c("meanBPpre","ExpCode","ExpType","GSlevel","Fictitious")
  P239.t.test(mattsDF,1,5,c("fact1","fact2"))

#Similarly, if there were more than two factors which appear more than
#once, we could perform all possible pairwaise T-tests using the function
#P239pairwise.

#A convenient method has been provided to produce boxplots:
  P239boxplot(mattsDF,"meanBPpre","ExpType")

#If a warning message is presented, that's OK.  The setting of colors was
#set specifically for the P239 experiments and will be generalized later.


#Dynamic Analysis
#================

#It can be instructive to look at the group mean dynamic behaviour of
#physiological variables.  ESIstat provides functions to perform this
#easily.  First, the data has to be syncronized and interpolated to a
#second-by-second basis.  This is done with:

  DynamicBP<-dynamicpts("mattsobj",GZtimes[,1:2],"HYS")

#Note that the first column of the second argument is used to syncronise
#the data).  The series of plots can be confusing at first.  First, the
#data from an insertion is plotted.  Then you are prompted to remove
#points.  Enter zero and return if there are no extreme outliers to
#remove.  Then, you must enter return again to see the interpolated data
#and the GZ signal.  The interface can be spruced up a little later.

#Next, fix the rownames of DynamicBP.  Since we are looking for group
#means, we should choose rownames which represent the factors we wish to
#group by.  Let's use fictitious ones:
  rownames(DynamicBP)<-c("GZdata","Timestamps",c("fact1","fact1","fact1",
  "fact2","fact2","fact2"))

#Note that DynamicBP has a special structure.  The first row is the GZdata,
#the second row is the timestamps, and the remaining rows are dynamic data
#for each insertion.  Now we can plot the group mean second-by-second data
#for different groups:

  dynplot(DynamicBP,c("fact1"))          #to see only fact1
  dynplot(DynamicBP,c("fact2"))          #to see only fact2
  dynplot(DynamicBP,c("fact1","fact2")) #to see both fact1 and fact2
```

We may wish to see "delta" values. That is, the difference between each second by second value from the mean of the resting value. This is done by using the dynplotdelta function which takes, as an argument, the dynamic data, calculates the delta values and produces a plot. The resulting delta data frame can be assigned to a variable if desired:

```
  DeltaBP<-dynplotdelta(DynamicBP,4,c("fact1","fact2"))
```

Note, that the 4 is the length of the resting period used for baseline (calculated at the beginning of the data sets). Also note, that the y-axis of the graph has been set to a poor #scale (which was used for ECG signal), so the graph looks awful. This will be fixed.

```
#Signal Handling and Processing Functions
#=========================================

#Several function have been included to perform various signal
#processing and data handling tasks. They are described below.
```

To get a channel of data from a data file, TSTAT is called via TSTATgetdata. Note that the data returned is for the current insertion in TSTAT. This can be changed with the generic call to TSTAT and "select". callTSTAT is normally used internally in ESIstat and shouldn't be needed for much more than selecting the current insertion.

```
  callTSTAT("mattsobj","select", "S5e5")

#where "S5e5" can be replaced with the desired insertion (e.g.,
"S5e1", "S5e11", etc.)

  ECG<-TSTATgetdata("mattsobj","Whole","ECG")
  ECGP<-TSTATgetdata("mattsobj","Whole","ECGP")

#The ECG peaks were calculated beforehand and saved in the datatrac file.

  plot(ECG,type="l",ylim=c(-1,1),xlim=c(0,70))
  par(new=T)
  plot(ECGP,type="p",ylim=c(-1,1),xlim=c(0,70),col="red")

#Now, calculate the heart rate:
  HR<-ecgp2hr(ECGP)
  par(new=T)
  plot(HR,type="l",ylim=c(0,2),xlim=c(0,70),col="blue")

#Now, calculate the RR interval:

  RR<-ecgp2rr(ECGP)
  par(new=T)
  plot(RR,type="l",ylim=c(0,2),xlim=c(0,70),col="green")

#There are similar functions for blood pressure calculations:

  SYS<-TSTATgetdata("mattsobj","Whole","SYS")
  DIA<-TSTATgetdata("mattsobj","Whole","DIA")

#The systolic and diastolic peaks were calculated beforehand and
#saved in the datatrac file.

  plot(SYS,type="l",ylim=c(50,130),xlim=c(0,70))
  par(new=T)
  plot(ECGP,type="p",ylim=c(50,130),xlim=c(0,70),col="blue")

#Now, calculate the mean arterial pressure:

  MAP<-sysdia2map(SYS,DIA)
  par(new=T)
  plot(MAP,type="l",ylim=c(50,130),xlim=c(0,70),col="green")
```

```
#Now, calculate the pulse pressure:

  PP<-sysdia2pp(SYS,DIA)
  par(new=T)
  plot(PP,type="l",ylim=c(20,60),xlim=c(0,70),col="red")

#Now, calculate the Stiffness Index:

  SI<-sysdia2si(SYS,DIA)
  par(new=T)
  plot(SI,type="l",ylim=c(200,700),xlim=c(0,70),col="purple")


#Miscellaneous Functions
#======================
```

The following few functions are very simple and straightforward but are presented here for completeness. "last" simply returns the last value in a vector. It was useful to call in a loop when looking for the last value of a parameter (eg. Heart rate) before a tilt. It can take and vector.

```
  last(c(1,3,4,2))  #would return 2
```

calcslope calculates the slope of a line represented by two vectors (x values and y values). It just calls R's lm() function.

```
calcslope(c(20,40,60,80),c(1,2,3,4))
#would return a linear model with slope of 20 and intercept of zero.
```

firstquart simply return the first quartile of a vector. This was considered as an alternate to a "minimum" when outliers could effect the result.

```
firstquart(c(0,10,20,30,40,50,60,70,80,90,100)) #would return 25
```

#Vdecimate is used to decimate a vector by a given factor.

```
Vdecimate(c(1,2,3,4,5,6,7,8,9),2) #would return 1,3,5,7,9
```

Certain functions are used internally in ESIstat and shouldn't be typically needed to the user, such as verifyDFC

ESIstat Tutorial 2: Statistical Analysis for Protocol L-239 & AsMA Paper

The following R objects constitute the main sets of data used to produce the recent abstract on the Protocol 239 (P239) Tilt-table Experiments. The objects are saved in a binary R workspace named P239Robjects and can be loaded into an R session with the command:

```
load("PathToP239Robjects")
```

for example, if the file is located under d:\temp\ the command would be:

```
load("d:\\temp\\P239Robjects")
```

Note the double \\ required in path names.

An ascii version is also available under P239Robjects.txt

newP239 this is the TSTAT object containing all of the runs in the experiment. In order to use this object using either TSTAT or ESIstat commands, the data files must be located in the proper directory. The directory is /mnt/D/P239/Temp in a Linux environment.

firstDF2	a dataframe containing 5 second Pre-Tilt "last" and 5 second Post-tilt minimum values for Systolic Blood Pressure (SBP), Diastolic Blood Pressure (DBP),

	Stiffness Index (SI), and Heart Rate (HR). The differences (deltas) are also included.
firstDFmeans	a dataframe similar to firstDF containing 15 second Pre-tilt mean and 5 second post-tilt mean values
hrDF	a similar dataframe which is dedicated to heart rate. The columns contain the 5 second Pre-Tilt mean and head-down-tilt minimum heart rate values.
ExpRuns	a vector of factors containing the Experiment code names for each of the insertions in firstDF2
fullnewGZpts	The times of GZ transitions for the insertions in firstDF2.
relaxGZtimes	the times used for GZ transition points when there was no tilt.
newdeltas	a dataframe combining information in firstDF2 and firstDFmeans. This dataframe contains the delta values calculated as the difference between pre-tilt mean and post-tilt minimum for SBP, DBP, RR interval. This combination appeared to be the most informative. Corresponding heart rate data appears in newdeltaHR.
newdeltaHR	similar data as in newdeltas, but containing heartbeat rate.

Dynamic data frames are also saved as R objects. For each of SBP, DBP, HR, and SI the dynamic data is calculated using dynamicPlot function and the delta values from a 5 second rest period pre-tilt are calculated using dyndeltaplot. The corresponding data frames are:

SYSdyndata
SYSdyndelta
HRdyndata
HRdyndelta
SIdyndata
SIdyndelta

Selection of Indices for Data Frames
====================================

To perform a statistical analysis on any given signal from an experiment the first step is to decide on the desired indices. In other words, what aspect of the signal is of interest. Specific indices could be the mean of the signal, the maximum of the signal during a given time interval, the standard deviation of the signal, etc. Two main approaches were used in analyzing Protocol L-239 experiments. Both approaches focused on the time regions including the pre-tilt resting value of the signal, and the post-tilt values since. The pre-tilt values are used as a baseline while the post-tilt values indicate the effectiveness of the protection strategy.

1) For each physiological variable (eg. SBP, DBP, Heart Rate) the LAST GOOD value in the 5 seconds before tilt was selected. It was reasoned that the LAST value would best indicate the physiological state of the subject before tilting. For the post-tilt region the MINIMUM value in the 5 seconds starting at HDT-HUT transition was selected. It was found by experience that at least 5 seconds post-tilt is required to discern between protection effects. This is the strategy used to generate firstDF2. The graphical exploration of this data frame was handed out at meetings with the SA (Bill Fraser) and appears in the file "d:\linuxWork\p239results\FirstTry.zip" on PCESI.

2) For each physiological variable (eg. SBP, DBP, Heart Rate) the MEAN value in the 15 seconds before tilt was selected. Here it was reasoned that selecting only one value (ie the LAST value) results in too much noise to generate reliable results. It was expected that a MEAN over 15 seconds would yield more consistent results. For the post-tilt region the MEAN value in the 5 seconds starting at HDT-HUT transition was selected. Again, it was reasoned that the MEAN post-tilt value would yield consistent results. This is the strategy used to generate firstDFmeans. The graphical exploration of this data frame was handed out at meetings with the SA (Bill Fraser) and appears in the file "d:\linuxWork\p239results\SecondTry.zip" on PCESI.

In the end, as the following presentation will show, it was found that a combination of the two approaches would give the best results. The pre-tilt MEAN value yields consistent baseline values and the post-tilt MINIMUM value best displays the effectiveness of the control strategy. To summarize the data, we take the difference (delta) between the pre MEAN and post MAXIMUM values. The result is stored in the dataframe dubbed newdeltas. Note that to calculate the MINIMUM heart rate, the MAXIMUM RR interval must be used, as shown in firstDF2. The heart rate values appear in the dataframe dubbed hrDF.

Commands for creating the aforementioned data frames:

===

Assume that a valid TSTAT object, named newP239, is available, with all 207 insertions added and the "Whole" windows identified (see Tutorial 1). The set of commands defined below will generate and analyze the data as was done to prepare the recent AsMA abstract for the L-239 experiments and the results presented in this report.

```
##Calculate the mean SYS in the 15 seconds before tilt
newdeltas<-Dfapplyfun("calcDFC",cbind(fullnewGZpts[,1]-15,fullnewGZpts[,2]),
"newP239","Whole","HYS",1,207,"mean")

newdeltas<-as.data.frame(newdeltas)

##Calculate the min SYS in the 5 seconds starting at HDT-HUT
transition
```

```
temp1<-Dfapplyfun("calcDFC",cbind(fullnewGZpts[,3],fullnewGZpts[,3]+5),
"newP39","Whole","HYS",1,207,"min")

##Calculate the delta

temp2<-temp1-firstDF2[,1]

##Set newdeltas to be a data frame

newdeltas<-as.data.frame(newdeltas)

##combine the previous 3 columns of data into a dataframe and name the
columns

newdeltas<-cbind(newdeltas,temp1,temp2,ExpRuns)

names(newdeltas)<-c("SYSmeanpre","SYSminpost","DeltaSYS","ExpType")
```

One way to conduct a t-test between experiment types is to extract the data from two types of runs into new variables.

```
##Extract the delta values for control runs.  This could also be
done using P239subset function.  Note that column 4 contains the
Experiment code and column 3 contains the delta values

temp1<-newdeltas[newdeltas[,4]=="Control1-1",3]

##Extract the delta values for the protected runs. This could also
```

be done using P239subset function.

```
temp2<-newdeltas[newdeltas[,4]=="TPP-A4-3",3]
```

```
##Now perform the t.test
```

```
t.test(temp1,temp2)
```

A simpler way to perform the t.test in one line is to use the P239.t.test function as follows:

```
P239.t.test(newdeltas,1,4,c("Control1-1","TPP-A4-3"))
```

Note that this function always uses the two.sided t.test. For the AsMA paper, it has been found that the paired t.test was more appropriate. In this case, either modify the P239.t.test function, or use the first method with a third argument alternative = "two.sided"

ANOVA Tests
===========

Having calculated the desired indices, such as deltaSBP, we can now begin to use the analysis of variance method to determine which protection strategies are significantly different in a statistical sense. The ANOVA test is carried out using the R "aov" function and specifying the terms to use and the source of the data. For example, to see if the DeltaSBP is significantly different between run types, the command is:

```
summary(aov(DeltaSYS~ExpType,data=newdeltas))
```

Note that the terms in the formula correspond to the names of the columns in the dataframe.

The resulting printout will display the summary of the ANOVA test. A P values < 0.05 is generally accepted as indicating significance. Of course, in this example we would expect significant differences between at least some Experiment types since a wide variety of experiments are included, from relax runs to unprotected tilts, to protected runs with no tilt. A more meaningful test would be to focus in on a few experiment types. This is done by extracting a subset of the data frame to analyze. One way to do this is with the P239subset function as follows:

```
PPBvsControl<-P239subset(newdeltas,4,c("Control1-1","TPP-A4-1",
"TPP-A4-2","TPP-A4-3"))
```

```
summary(aov(DeltaSYS~ExpType,data=PPBvsControl))
```

Alternatively, this could be combined into one line:

```
summary(aov(DeltaSYS~ExpType,data=P239subset(newdeltas,4,c("Control1-1",
"TPP-A4-1","TPP-A4-2","TPP-A4-3"))))
```

Dynamic Analysis Examples

The dynamic data frames mentioned earlier are very useful for visualizing the POPULATION-WIDE trend for a given protection strategy. By plotting various pairs of protection runs and noting how and when they differ, one can determine where to best focus efforts on statistical analysis. The dynamic data frames are calculated as follows:

```
##Calculate dynamic points
SYSdyndata<-dynamicpts("newP239",fullnewGZpts[,1:2],"HYS") Note
that 1-st column of the second argument is used for
synchronization.

##Assign proper rownames
rownames(SYSdyndata)<-ExpRuns

##Plot one or more runs of interest. For more than one run,
include extra terms in the second argument
dynplot(SYSdyndata,c("Control1-1"))

#Calculate delta values based on the five seconds of rest before
the tilt. The dynplotdelta calculates the delta values for all
run types and displays the ones of interest at the same time.

SYSdyndelta<-dynplotdelta(SYSdyndata,5,c("Control1-1"))
```

Follow similar methods for HR and SI.

Now, note that it is easy to select all dynamic data for a certain time period, say HDT, and compare between Experiment type. Simply extract the columns of interest from the dynamic data frames.

```
##Examine the first two rows of the dynamic dataframe to find the
columns for HDT.

SYSdyndata[1:3,]

##Examining the row of tilt data, it can be seen that columns 10
to 24 are the HDT period

hdtSYDdyndata<-SYSdyndata[,10:24]
```

now, to perform a t.test based on this data, the columns must be stacked one above another using the stackandbox function. This function returns a stacked dataframe and displays a boxplot of the data, plotted against the values in the rownames of the dataframe.

```
stackedhdtSYSdyndata<-stackandbox(hdtSYSdyndata)
P239.t.test(stackedSYSdyndata,1,2,c("Control1-1","TPP-A4-3"))
```

Finally, one may wish to group several types of experiments together to strengthen the statistical power. For example, up to the HDT-HUT transition, experiments which include only PPB during

HDT are the same as experiments which include PPB during HDT and G-suit after tilt. If you are considering only the HDT data for analysis, you may wish to group these experiments together. This can be done with the groupdata function:

```
PPBvsNOPPB<-groupdata(SYSdyndata,"rownames",c("TPP-A4-1","TPP-A2",
"TPP-A4-2","TPP-A4-3","TPG-A47","TPG-A27"),
c("Control1-1","Control1-2","TGS-A7","TPG-A77-1","TPG-A77-2"),
c("PPBduringHDT","NoPPBduringHDT"))
```

Note that the second argument of groupdata could be either "rownames" or a number that would represent the number of a column to which grouping should be applied.

Note also that the experiments in the third and fourth arguments will be classified as PPBdurngHDT and NoPPBduringHDT respectively. Now an analysis could be performed to examine the effect of PPB during HDT. There are, of course, other ways to accomplish this.

This grouping method was used to determine the effect of PPB during HDT by combining runs into one of two groups: PPB during HDT (runs TPP-A4-1, TPP-A2, TPP-A4-2, TPP-A4-3, TPG-A47, TPG-a27) and No PPB durng HDT (Control1-2, TGS-A7, TPG-A77-1, TPG-A77-2, Control1-2). The results were plotted showing the mean and standard error at each second using dynplot.

Similarly, the effect of wearing a mask without PPB during HDT was investigated by combining data into two groups: Mask without PPB during HDT (TPG-A77-1 and TPG-A77-2) and No mask at all during HDT (Control1-1, Control2, TGS-A7, Control1-2). All of these results were handed out at meetings with the SA (Bill Fraser) and appear in the file "d:\LinuxWork\P239results\PPBandMaskEffect.zip" on PCESI.

Analysis for AsMA Abstract

ANOVA analysis was first performed to look for differences between protection strategies. An excerpt from our original abstract ("d:\LinuxWork\p239abstract2.txt")is as follows:

Repeated measures ANOVA analyses showed a significant difference in delta SBP (P=0.03135) and delta HR (P=0.0036) between different runs with experimental candidate protection schedules.

This can be recreated with the following command:

```
temp3<-P239subset(newdeltas,4,c("Control1-1","TPP-A4-3","TPP-A2","TGS-A7",
"TPG-A47","TPG-A27","TPG-A77-1"))
```

```
summary(aov(SBPdeltameanmin~ExpRuns+Error(Subjects),data=temp3))
```

Note that the correction for Repeated Measures ANOVA is accounted for by the Error(Subjects) term. There is some discrepancy about this in the literature.

The heart rate ANOVA is performed in a similar manner.

```
#Compare Control1-1 to TPP-A2 to determine effect of PPB during
HDT and Transition

temp1<-P239subset(newdeltas,4,"Control1-1")
temp2<-P239subset(newdeltas,4,"TPP-A2")

#t-test for Delta Systolic Blood Pressure
 t.test(temp1[,1],temp2[,1],paired=T)
#t-test for Delta Diastolic Blood Pressure
 t.test(temp1[,2],temp2[,2],paired=T)
#t-test for Delta RR interval
 t.test(temp1[,1],temp2[,1],paired=T)
```

Perform similar comparisons between other runs, substituting the code name in the second argument of P239 subset. Paired comparisons made were:

TGS-A7 and TPG-A47 to compare additional benefit of PPB during HDT when G-suit is applied upon the tilt from HD to HU.

TGS-A7 and TPG-A27 to compare additional benefit of PPB during HDT and throughout transition when Gsuit is applied

TPG-A77-1 and TPG-A27 to compare the benefits of applying PPB during HDT when both PPB and G-suit are applied upon HD-to-HU tilt

Appendix D:

Experimental Protocol L- 239

September 8, 1999

DCIEM Experimental Protocol L- 239

Evaluation of a Subject's Physiological Responses to Non-linear G-suit, Positive Pressure Breathing Schedules, and Straining during Low Intensity Push-Pull Transitions

Principal Investigator: William Fraser, H/SAILSS Team

Co-Investigators: Vahid Askari, Ziren Lu, Alex Kapps (Engineering Services Inc., Toronto); John Yeow (University of Toronto)

Medical Advisor: Dr. G. Gray.

1.0 Background

Pilots must perform flight operations involving cycles of ±Gz maneuver. It is known that pilots' +Gz tolerance is reduced significantly when a +Gz exposure has been preceded by a -Gz period. This phenomenon is called 'Push-Pull' Effect [1].

The intensity of negative Gz in aerial combat maneuvers can reach -2.0 to -2.5. However, the DCIEM Centrifuge can only generate positive Gz, whereas a Tilt Table generates accelerations ranging from at most -1Gz to +1Gz. The intention of this set of experiments is to develop an Anti-G Protection Expert System (AGPES) for a typical subject during low intensity push-pull transitions. This experiment is a continuation of previous experiments. The intention of the previous set of experiments (Protocol#-L206) was to test the utilization of various G-suit pressure and positive pressure breathing (PPB) profiles combined with negative tilt for inducing and increasing negative acceleration related physiological effects [9] It is known that PPB with G-suit inflation will cause a number of physiological responses in a subject's cardiovascular functions that are similar to those resulting from an exposure to negative Gz, such as: (i) an increase in venous return from lower part to upper part of a human body; (ii) higher than normal blood pressure at head level; and (iii) a decrease in heart rate [2, 3, 4]. In addition, a study will be undertaken to examine the effect of a set of pressure profiles in protecting against the push-pull effect.

2.0 Aims

The purpose of this study is to:

a. Test and evaluate an Anti-G Protection Expert System to enhance negative Gz effects at –1Gz and +1Gz,

b. Test candidate protection techniques against Push-Pull effects.

3.0 *Experimental Design and Protocols*

3.1 General Experimental Design

The experiment will be performed on Tilt-Table with seated subjects wearing G-suit (STING) with CF PPB jerkin and PPB capable masks. Different profiles of Tilt-Table motion with different amplitudes and timings of G-suit pressures and PPB will be applied Maximum Tilt-Table rotation of 135° (Head Down Tilt "HDT") is being used to simulate -1Gz that is not exactly -1GZ (~-0.7Gz). Each subject will complete 25 runs (control runs & tilt-table runs) in 3 hours.

Subjects will be recruited from DCIEM A-team members and others medically qualified for PPB and tilt-table exposures. All new subjects will be required to attend a one-hour training session on a separate day prior to the actual experiments during which the subject will be introduced to the laboratory and the biomedical monitoring equipment and a set of tilt-table rotations with subject wearing a non-pressurised G-suit and mask-jerkin ensemble The subjects must agree: (i) to not consume any alcoholic beverages within 24 hrs of each series of experiments; (ii) to have a restful night evening the night before a test; (iii) to perform no heavy exercise on the day of a test; and (iv) to abstain from smoking, coffee, cola beverages and chocolate in the 3 hrs period before a test, and a meal in the hour preceding it. Any prescription drugs must be approved by both the Principle Investigator and the Medical Advisor. A blood donation less than 30 days in advance of commencement of experimentation is a contraindication to participation in the study.

All subjects will be trained in performing positive pressure breathing up to a maximum of 50 mmHg. The training will include a comprehensive briefing on the test equipment, the experimental procedures and the physiological effects of PPB.

3.2 Subjects

Between 8-10 male and female volunteers, between 18 and 50 years of age, will be recruited for this experiment. Each subject will be provided with an information package which outlines the experimental protocol and which also contains the consent forms to be signed by the subjects. Participation will be open to civilian and military members. Subjects will be screened for PPB exposure by a DCIEM physician. Investigations will include ECG (ElectroCardioGraph), CXR (Chest X-Ray), and detailed PFT (Pulmonary Function Test). Before the experiment subjects will be monitored if there is any interim change in their medical status (Temperature (oral) > 37^2, or Blood Pressure > 140/90). Subjects with positive findings will send to a DCIEM physician. Subjects will be recruited by personal contact

3.3 Test Procedures

Subjects will complete six continuous runs consisting of relaxed, pre-planned tilt-table motion (Figure 1), G-suit, PPB, G-suit & PPB schedules, and straining & PPB.

Figure 1: Tilt table rotation; HUT: Head Up Tilt (15 °), HDT: Head Down Tilt (135 °)

Run 1: Relaxed run (Subject seated on Tilt-table "15 degree" wearing a non-pressurised G-suit and mask-jerkin ensemble)

No variable manipulated in this experiment. The subject's physiological parameters will be measured (e.g. blood pressure, ECG) during a period of three minutes seating on the tilt-table at 15°.

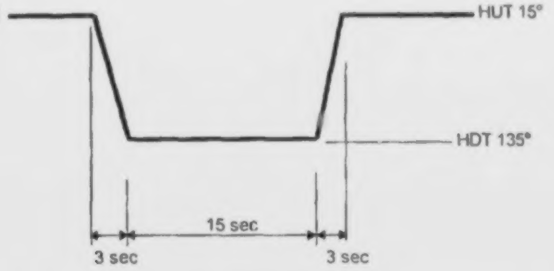
Run 2: Control run (Tilt-table rotation with subject wearing a non-pressurised G-suit and mask-jerkin ensemble)

The variables manipulated in this experiment are (refer to Figures 2, 3):

a.	Tilt-table onset rate:	45° /sec.
b	tilt-table amplitude:	90 ° & 135 ° .
c.	plateau duration:	120 & 15 sec

Total number of different profiles: 3

Tilt-Table Profile 1



Tilt-Table Profile 2

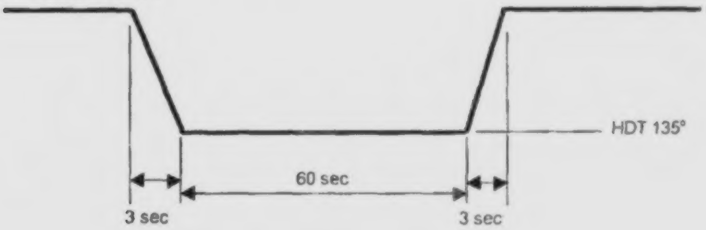


Figure 2: Tilt table control run (profile 1, 2)

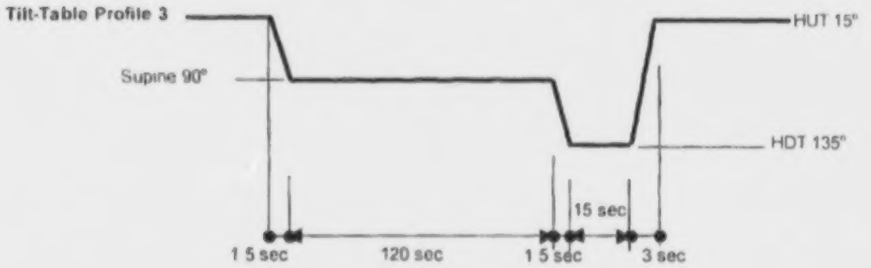

Tilt-Table Profile 3
Supine 90°
HUT 15°
HDT 135°
15 sec
1 5 sec
120 sec
1 5 sec
3 sec

Figure 3: Tilt table control run (profile 2)

Table 1: Tilt-Table Profiles for Run 1, 2

	onset time (sec)	amplitude (degree)	plateau duration (sec)
1	3	135	15
2	3	135	60
3	3	90 &135	120 & 15

Run 3: Combined tilt-table rotation and G-suit pressures

The parameters of Tilt-Table motion profiles are the same as in Run 2 (tilt-table profile 1) The G-suit variables manipulated in this experiment are (see also Figures 4, 6 and Table 2):

a. onset time:	From HDT transition (90 to to HUT (see Figure 5).
b. amplitude:	4 psi.
c plateau duration:	25 sec
d G-valve:	SAMCAV

Total number of different profiles is: 3

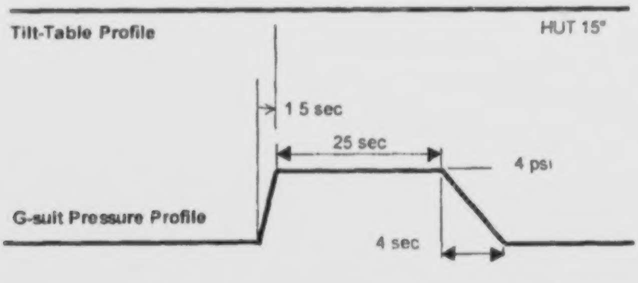

Tilt-Table Profile
HUT 15°
1 5 sec
25 sec
4 psi
G-suit Pressure Profile
4 sec

Figure 4: G-suit profiles for Run 3 (control runs)

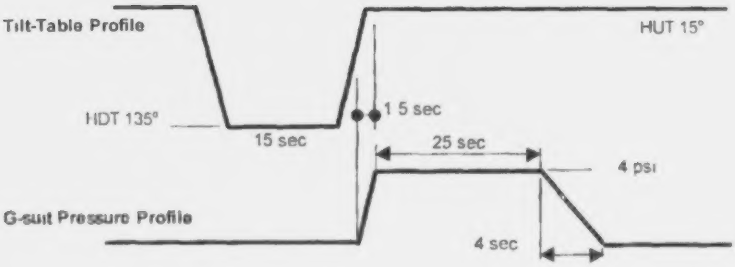

Tilt-Table Profile
HUT 15°
HDT 135°
15 sec
1.5 sec
25 sec
4 psi
G-suit Pressure Profile
4 sec

Figure 5: G-suit and tilt-table profiles for Run 3

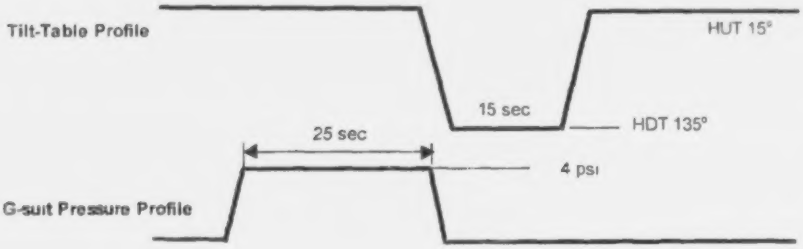

Tilt-Table Profile
HUT 15°
15 sec
HDT 135°
25 sec
4 psi
G-suit Pressure Profile

Figure 6: G-suit and tilt-table profiles for Run 3

Table 2: Tilt-table and G Suit Profiles for Run 3

	Tilt-Table			G-suit			Comments
No.	onset (sec)	amplitude (degree)	Plateau (sec)	plateau (sec)	onset (sec)	amplitude (psi)	
1	N/A	N/A	N/A	25	1.5	4	SAMCAV valve
2	3	135	15	25	1.5	4	SAMCAV valve
3	3	135	15	25	1.5	4	SAMCAV valve

Run 4: Combined tilt-table rotation and positive pressure breathing (PPB)

The parameters of Tilt-table motion profiles are the same as in Run 2 (tilt-table profile 1). The PPB variables manipulated in this experiment are (seeing also Figures 7-9 and Table 3):

a. onset time:	transition from HUT (90 degree) to HDT (see Figure 7).
b. Amplitude	20, 30 mmHg.
c. Plateau duration.	10, 25 & 50 sec.
d. G-valve:	SAMCAV.

Total number of different profiles is: 6

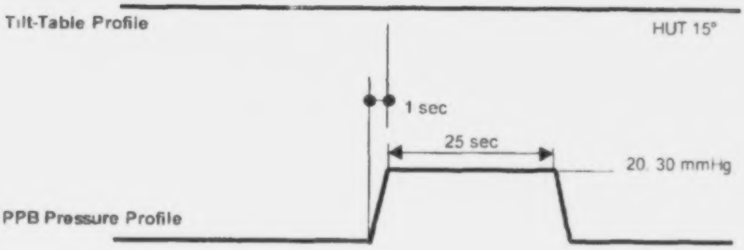

Tilt-Table Profile
HUT 15°
1 sec
25 sec
20, 30 mmHg
PPB Pressure Profile

Figure 7: PPB profile for Run 4 (control runs)

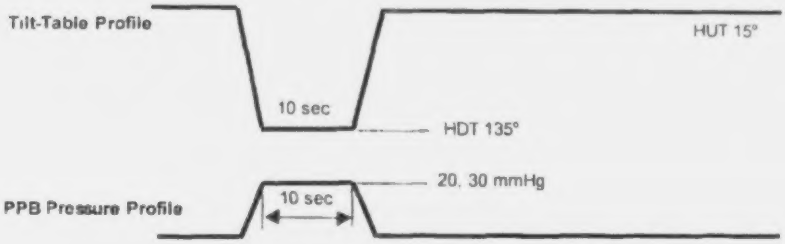

Tilt-Table Profile
HUT 15°
10 sec
HDT 135°
20, 30 mmHg
10 sec
PPB Pressure Profile

Figure 8: PPB and tilt-table profiles for Run 4

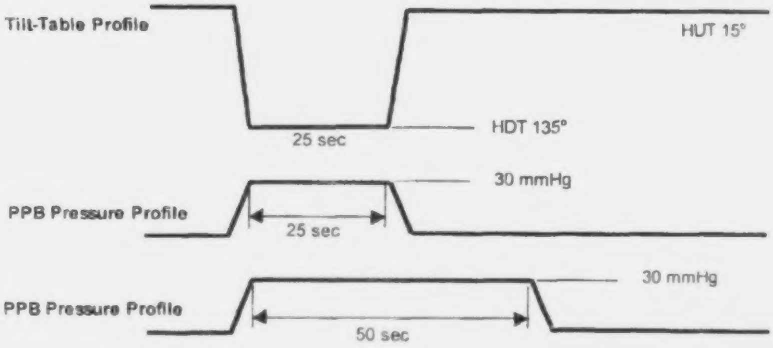

Tilt-Table Profile
HUT 15°
HDT 135°
25 sec
30 mmHg
PPB Pressure Profile
25 sec
30 mmHg
PPB Pressure Profile
50 sec

Figure 9: PPB and tilt-table profiles for Run 4

Table 3: Tilt-table and PPB Profiles for Run 4

	Tilt-Table			PPB			Comments
No.	Onset (sec)	amplitude (degree)	Plateau (sec)	plateau (sec)	onset (sec)	amplitude (mmHg)	
1	N/A	N/A	N/A	25	1	20	SAMCAV valve
2	N/A	N/A	N/A	25	1	30	SAMCAV valve
3	3	135	10	10	1	20	SAMCAV valve
4	3	135	10	10	1	30	SAMCAV valve
5	3	135	25	25	1	30	SAMCAV valve
6	3	135	25	50	1	30	SAMCAV valve

Run 5: Combined Tilt-Table Rotation, G-Suit Inflation and PPB Pressures

The profiles of Tilt-Table rotation, G-Suit, and PPB pressures will be applied in combinations as shown in Figures 10-13 and Table 4.

a. onset time:	transition from HUT (90 degree) to HDT (see Figures 10-14)
b. Amplitude (PPB):	30, 50 mmHg.
c. Amplitude (G-suit):	4, 5 25 psi.
d. Plateau duration:	15, 25 & 40 sec.
e. G-valve:	SAMCAV
Total number of different profiles is.	9

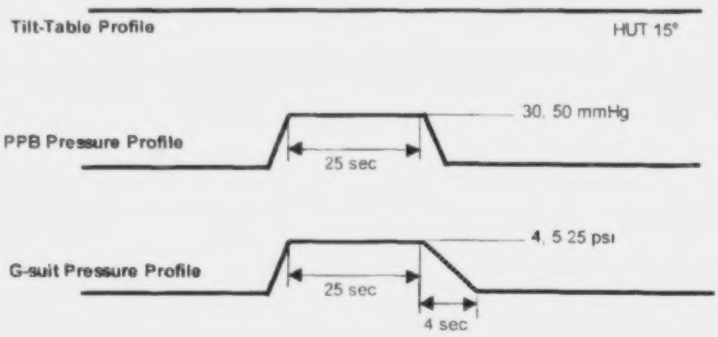

Tilt-Table Profile
HUT 15°
30, 50 mmHg
PPB Pressure Profile
25 sec
4, 5 25 psi
G-suit Pressure Profile
25 sec
4 sec

Figure 10: PPB and G-suit pressure profiles for Run 5 (Control run)

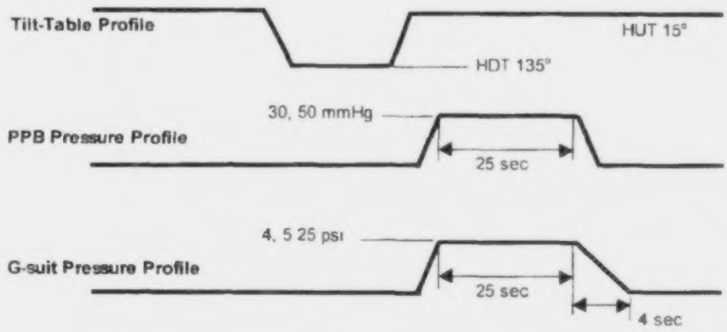

Tilt-Table Profile
HUT 15°
HDT 135°
30, 50 mmHg
PPB Pressure Profile
25 sec
4, 5 25 psi
G-suit Pressure Profile
25 sec
4 sec

Figure 11: Tilt-table rotation, PPB, and G-suit pressure profiles for Run 5

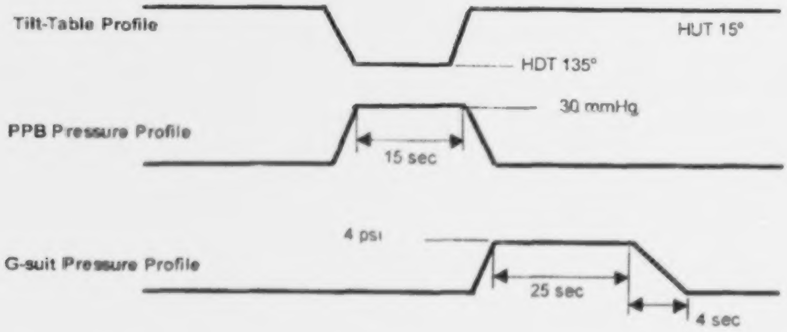

Tilt-Table Profile
HUT 15°
HDT 135°
PPB Pressure Profile
30 mmHg
15 sec
4 psi
G-suit Pressure Profile
25 sec
4 sec

Figure 12: Tilt-table rotation, PPB, and G-suit pressure profiles for Run 5

1.

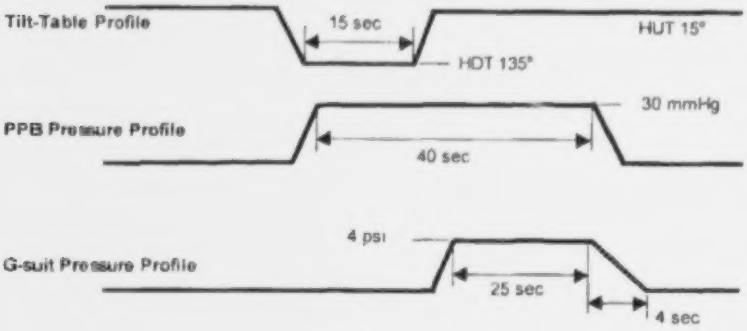

Tilt-Table Profile
15 sec
HUT 15°
HDT 135°
PPB Pressure Profile
30 mmHg
40 sec
4 psi
G-suit Pressure Profile
25 sec
4 sec

Figure 13: Tilt-table rotation, PPB, and G-suit pressure profiles for Run 5

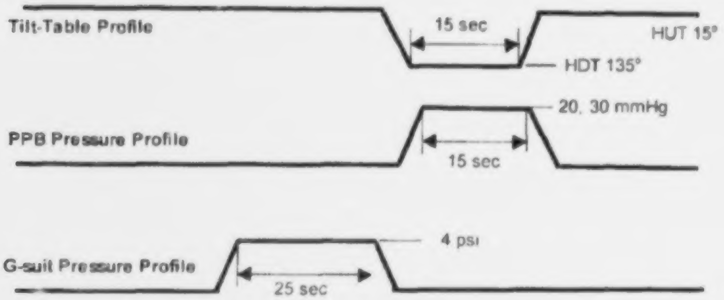

Tilt-Table Profile
15 sec
HUT 15°
HDT 135°
PPB Pressure Profile
20, 30 mmHg
15 sec
G-suit Pressure Profile
4 psi
25 sec

Figure 14: Tilt-table rotation, PPB, and G-suit pressure profiles for Run 5

Table 4: Tilt-Table, G-suit and PPB Profiles for Run 5

	Tilt-Table			G-suit			PPB			comments
	onset (sec)	Amp. (degree)	Plat (sec)	onset (sec)	Amp. (psi)	Plat. (sec)	onset (sec)	Amp. (mm Hg)	Plat (sec)	
1	N/A	N/A	N/A	2	4	25	1	30	25	SAMCAV
2	N/A	N/A	N/A	2	5.25	25	1	50	25	SAMCAV
3	3	135	15	2	4	25	1	30	25	SAMCAV
4	3	135	15	2	5.25	25	1	50	25	SAMCAV
5	3	135	15	2	4	25	1	30	15	SAMCAV
6	3	135	15	2	4	25	1	30	40	SAMCAV
7	3	135	15	2	4	25	1	30	25	SAMCAV
8	3	135	15	2	4	25	1	20	25	SAMCAV
9	3	135	15	2	4	25	1	30	25	SAMCAV

Run 6: Combined tilt-table rotation, straining and positive pressure breathing (PPB)

The parameters of Tilt-Table motion profiles and PPB profiles are the same as the profiles in Run 4. There exists a straining parameter in this run (Valsalva (M-1) for 15 sec.) The combinations of straining, tilt-table rotation and PPB are shown in Figures 15-17 and Table 5.

a.	onset time:	transition from HUT (90 degree) to HDT (see Figure 15).
b.	Amplitude (PPB):	30 mmHg.
c.	Plateau duration:	25 sec.
d.	Straining	Valsalva (M-1) for 15 sec
e.	G-valve:	SAMCAV.

Total number of different profiles is: 3

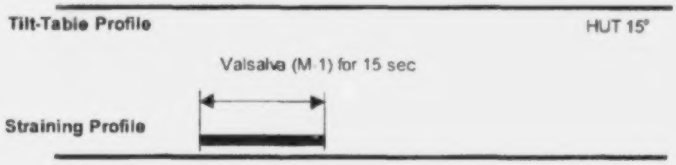


Figure 15: Straining control run for Run 6

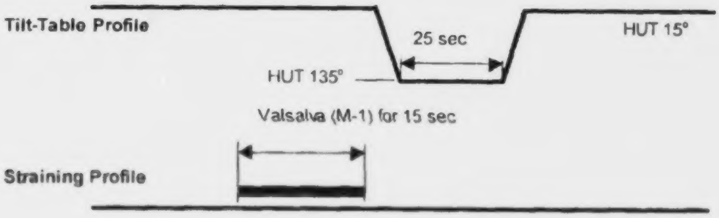

Tilt-Table Profile
25 sec
HUT 15°
HUT 135°
Valsalva (M-1) for 15 sec
Straining Profile

Figure 16: Straining and tilt-table rotation for Run 6

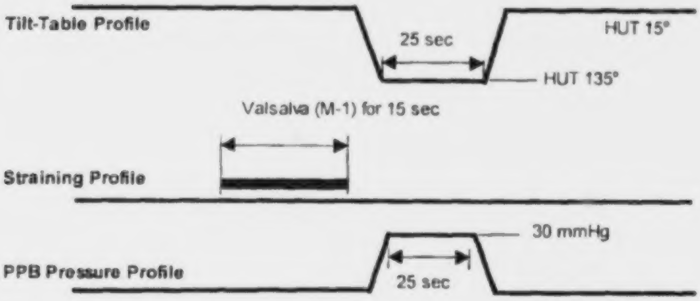

Tilt-Table Profile
25 sec
HUT 15°
HUT 135°
Valsalva (M-1) for 15 sec
Straining Profile
30 mmHg
PPB Pressure Profile
25 sec

Figure 17: Straining, PPB profile and tilt-table rotation for Run 6

Table 5: Tilt-Table, Straining and PPB Profiles for Run 6

	Tilt-Table			Straining (Valsalva, M-1)			PPB			comments
	onset (sec)	Amp. (degree)	Plat. (sec)	Dur (sec)	Strain (period)	Exhale (sec)	onset (sec)	Amp. (mm Hg)	Plat. (sec)	
1	N/A	N/A	N/A	15	3	1	N/A	N/A	N/A	SAMCAV
2	3	135	25	15	3	1	N/A	N/A	N/A	SAMCAV
3	3	135	25	15	3	1	1	30	25	SAMCAV

3.4 Summary of Test Procedures

Test site:	Tilt-Table Laboratory
Number of runs for each subject:	6 (including control runs)
Number of different profiles in each run, respectively:	2, 3, 3, 6, 9, and 3
Maximum +Gz	+1G
Maximum -Gz and duration:	-0.71G, 25 sec.
Maximum G-suit pressure and duration (HUT):	5 25 psi, 25 sec.
Maximum PPB and duration (HUT):	50 mm Hg, 25 sec.
Maximum G-suit pressure and duration (HDT)	0 psi.
Maximum PPB and duration (HDT).	30 mm Hg, 25 sec.
Total time for each subject:	3.0 hours.

There will be a 10-minute rest between runs.

4.0 Instrumentation

Physiological measurements will assess transient cardiovascular and cerebrovascular events These measures will be recorded on a continuous basis. All of the techniques utilised (Table 6) are non-invasive·

Table 5: Physiological Measurements, Techniques and Instrumentation

Measurement	Technique	Instrument
heart rate	electrocardiograph	Cardio-line Digital ECG
stroke volume/cardiac output, contractility	impedance cardiograph	Minnesota Impedance Cardiograph (IC)
systolic, diastolic, mean blood pressure	arterial volume clamp plethysmography	Finapres™ and/or Portapres™
blood flow and peripheral venous pooling	a impedance plethysmography b. venous occlusion plethysmography	a. THRIM™ 4-channel bioimpedance monitor, b. Hockenson™ venous occlusion plethysmograph
middle cerebral artery blood flow velocity	transcranial Doppler sonography	Medisonics™ trans-cranial Doppler unit
peripheral blood flow, pooling, venous outflow	arterial and venous Doppler sonography	Biosound™ peripheral vascular Doppler imaging unit
head-level blood content/pulsatilit	optical plethysmography	ear opacity plethysmograph
cerebral oxygenation status	near-infrared spectroscopy	Nirascope® near-infrared spectrograph unit
skeletal muscle activity	electromyography	electromyography
blood $C0_2$, $C0_2$ pressure	transcutaneous membrane plethysmography	Radiometer-Copenhagen™ transcutaneous PC02/P02 oximeter
% blood saturation with haemoglobin	pulse oximetry transcutaneous	Radiometer-Copenhagen™ pulse oximeter
end-tidal alveolar $C0_2$ pressure	open-circuit pulmonary gas analysis	carbon dioxide analyser

5.0 Medical Monitoring

A DCIEM medical officer will be on-call within the building to attend to medical complications and emergencies if they should arise. Resuscitation equipment and appropriate drugs are available on site to deal with life-threatening medical emergencies.

Although the subjects are only exposed to ±1Gz, the relative physiological stress when transition from -Gz to +Gz and high G-suit and PPB pressures during -1Gz might be significant Thus, subjects may be susceptible to unexpected presyncopal symptoms, and even loss of consciousness

Consequently, the ECG, Ear opacity and blood pressure signals will be continuously monitored and displayed. These physiological measures can readily identify impending pre-syncope during orthostatic stress. During each experiment, these signals will be monitored by the investigator to determine if a run should be aborted. In case of a run abort due to physiological signs and symptoms, or upon request of the subject, the tilt table will be rapidly rotated to the 90-degree supine position In a medical emergency necessitating evacuation, the table will be rotated to the loading position (+10 degree). In the event of a power interruption or system failure during a tilt run and/or syncopal event, the table can be moved to any position passively. All DCIEM safety policies and procedures will be followed in case of a medical emergency.

6.0 Safety

6.1 Termination Criteria

PPB and G-suit pressures as well as tilt-table rotation will be terminated immediately upon:

a) the request of the subject;

b) if the subject presses the hand-held emergency toggle switch;

c) if the subject develops any untoward arrhythmia including ventricular tachycardia,, atrial tachycardia, ventricular ectopic beats > 10/min or a run of 3 or more premature ventricular contractions (PVCs);

d) if the subject has any symptoms of syncope such as feeling of coldness, sweating, nausea or facial pallor;

e) the subject loses consciousness;

f) any other reason identified by the investigators (e.g. any untoward arrhythmia (bradycardia with HR < 30 bpm).

Since pre-syncopal symptoms are possible in most of these experiments, termination criteria will apply in all cases

(a) In the case of abort from the +135° HDT position:

Head-down tilt (HDT) is characterised by a slowing of heart rate, with some sinus arrhythmia and lengthened PR interval. These responses are expected, and will not constitute sufficient reason for aborting a run. However, return of the tilt table to the horizontal position (supine) will be occur immediately if the subject develops any untoward arrhythmia (bradycardia with HR < 30 bpm, asystole, a run of PVC's), chest pain, dyspnea, nausea, headache, tingling, or dizziness. In some individuals with incompetent duodenal sphincters, there is a risk of stomach content reflux, causing potential aspiration when in the HDT position. Although this risk is small, medical suction equipment will be available in this event.

The subject may abort a run and end the experiment at any time by request, or by use of the hand-held emergency toggle switch. However, if untoward signs /symptoms arise, the decision to continue with the testing depends upon the investigator and if required, an on-call medical officer.

(b) In the case of abort from the +15° HUT position:

The previous -1Gz condition will in some individuals, significantly disturb the normally adequate blood pressure response to subsequent upright tilting, causing orthostatic symptoms. These symptoms will be similar to those experienced in the DCIEM human centrifuge, and they include transient loss of peripheral vision, greying of vision, complete loss of vision (black-out), and in some cases, complete loss of consciousness. The risks of these symptoms increases as the interventions (HDT angle) which exacerbate the push-pull effect are increased in severity A return of the tilt table to the horizontal (supine) position will be initiated if the subject displays the following symptoms

a. loss of ear opacity pulse for > 3 s.;

b loss of consciousness,

c. nausea;

d. on the command of the subject, or the observing medical officer or the run director;

e appearance of any untoward cardiac arrythmias.

7.0 *Risks and Hazards*

Minimisation of risks with the testing procedures is dependent on medical screening, continuous monitoring of physiological responses, prompt recognition of adverse physiological responses and symptoms, and adequate subject habituation and training. However, despite all precautions, certain risks remain. These risks may be broken down into three categories: (1) risk due to measurement procedures/interventions; (2) those due to mechanical events associated with the anti-G system device itself; and (3) those of inducing pre-syncopal symptoms, fainting and medical emergencies due to the PPB. The latter risk has been outlined in the previous section.

7.1 Risk of Measurement Procedures

All of the cardiovascular measurement techniques are non-invasive, and have been safely used in previous DCIEM investigations (Protocol L-163). A detailed description of the sensors are summarised in [1].

7.2 Risk due to Mechanical Devices

The electronic Anti-G valves have been used in previous DCIEM experiments and found to be safe, providing appropriate mechanical safeguards are taken [5, 6, 7]. They will be checked prior to each experiment. This device is designed and engineered to be safe under most possible scenarios, including power loss, runaway motor, and control system. However, there is a remote possibility that these systems, even with their built-in redundancies, could fail. There is a remote possibility of injury (if all the safety valves (two mechanical and two electronic safety valves), and software safety checking program fail at the same time) by applying too high pressure in G-suit and PPB The mask will be removed and the supply hoses will be immediately disconnected if these safety systems fail.

7.3 Risk due to PPB

The risks of PPB are well known, and documented elsewhere [2]. Minor risks include petechial hemorrhage (bruising), loss of central blood volume and fainting, breathing discomfort and facial discomfort. The pressure from the G-suit may cause limb tingling. More serious but rare complications can include fainting, pulmonary barotrauma resulting from inadequate thoracic counter-pressurization, with introduction of air into the arterial circulation, and cardiac arrhythmia. Minimization of these risks is ensured by proper medical screening, a redundant fail-safe pressure delivery system, and the wearing of properly fitted Anti-G-suits (the suit will be fitted by CF safety system tech) and thoracic counterpressure garments.

8. 0 Data Processing Method

Data for the experiment will be analysed using both repeated-measures Analysis of Variance, Trend Analysis, High-Order Frequency Analysis, Scattered Diagrams, Bootstrap Technique, and Fuzzy Trend Analysis. System identification analyses based on neural network and fuzzy logic will be performed in order to develop models of a subject's physiological responses.

9. 0 Benefits of Study

In this study, results of physiological responses to non-linear preplanned G-suit pressure and PPB schedules during subjects' exposure to Push-Pull maneuvers will be collected. Major expected benefits from the study are twofold:

(i) A technique to simulate and enhance negative Gz effects at -1Gz and +1Gz will be developed, which will enlarge the envelope of simulated Push-Pull levels achievable using DCIEM facilities (i.e., DCIEM centrifuge and tilt-table), and thereby greatly facilitate the development of Push-Pull countermeasures including the design of a new computer-controlled CF18 anti-G valve.

(ii) Preliminary models of protected (as compared to relaxed) subjects' physiological responses will be developed that will indicate candidate protection techniques against Push-Pull effects.

A significant expansion of information on protected subject's physiological responses to G-suit pressure and PPB during and immediately following negative acceleration is anticipated. Although the major purpose of this experiment is to simulate elevated levels of Push-Push transitions, this study will also facilitate the development of Anti-G pressure schedules aimed at reducing negative acceleration effects

10.0 Remuneration

The subjects for this study are entitled to stress allowance as described in Ackles. [8]

References

[1] L.S. Goodman, "Protocol #L-163: Tilt Table Simulation of Push-Pull Effect and Tilt Table Study, Squat-Stand Test and Risk Factor Investigation", DCIEM, January 1997

[2] M. Pecaric, F. Buick, "Protocol #L-28: Evaluation of PBG Schedules on +Gz-Intensity Tolerance", DCIEM, July 1992.

[3] L.S. Goodman, etc., "Cardiovascular Effects of Varying G-suit Pressure and Coverage During +1Gz Positive Pressure Breathing", Aviation, Space, and Environmental Medicine, pp. 829-836, September 1995.

[4] A.R.J. Prior, "The Optimization of a Positive Pressure Breathing System for Enhanced G Protection", Proceedings of AGARD, pp 14-1 to 14-11, April 1991

[5] A.A. Goldenberg, etc., "Design and Control of a New Anti-G Valve for High Energy Manoeuvring Aircraft - Design and Integration of an Experimental Research Facility for Anti-G Pressure Regulation and Positive Pressure Breathing Based on a New Valve System - DCIEM-Phase V", University of Toronto, RAL Report, May 1994.

[6] T. Gee, etc., "Blood Pressure Perturbation Experiment: Safety Systems and Testing", Aerospace Physiology Group, DCIEM, February 1995.

[7] T. Gee and W. Fraser, "Blood Pressure Perturbation Experiment Phase 1: Pilot Runs", Aerospace Physiology Group, DCIEM, April 1995.

[8] Ackles, K Memorandum, "Stress Allowance for DND Experimental Subjects", 7200-2 (BIO), 2 Dec 92 and 4 May 94

[9] Development of an Expert System for Automatic Generation and Adaptive Optimization of Push-Pull Countermeasures for Typical and Atypical Pilots- DCIEM Phase-X Final Report, March 1999.

DCIEM Human Ethics Consent Form Protocol # L-239r

Evaluation of a Subject's Physiological Responses to Non-linear G-suit and Positive Pressure Breathing Schedules during Low Intensity Push-Pull Transitions

Principal Investigator: William Fraser, H/ SAILSS Team
Co-Investigators: Vahid Askari, Ziren Lu, Alex Kapps (Engineering Services Inc., Toronto); John Yeow (University of Toronto)

Points of Contact: Protocol questions: William Fraser (Office 416 635 2064: Home: 905 272 3659)
Medical concerns: Dr. Gary Gray (Office 416 635 2015)

I,________________________________, __________________________ (address and telephone number) hereby volunteer to participate as a test subject in a DCIEM experiment to study the effects of nonlinear G-suit and breathing pressure schedules under low-intensity Push-Pull transitions. I have read the attached information package on the experimental protocol and description of risks, and have been informed to my satisfaction of all the risks associated with the experiment and consider these risks acceptable. In addition, I understand that this study, or any research, may involve risks that are currently unforeseen. I understand these risks. **For CF Members only:** I understand that I am considered to be on duty for disciplinary, administrative and Pension Act purposes during my participation in this experiment. **THIS DUTY STATUS HAS NO EFFECT ON MY RIGHT TO WITHDRAW FROM THIS EXPERIMENT AT ANY TIME I WISH AND I UNDERSTAND THAT NO ACTION WILL BE TAKEN AGAINST ME FOR EXCERCISING THIS RIGHT.**

All of the cardiovascular measurement techniques are non-invasive, and have been safely used in previous DCIEM investigations (Protocol L163). The electronic Anti-G valves have been used in previous DCIEM experiments and found to be safe, providing appropriate mechanical safeguards are taken [5-7]. This device is designed and engineered to be safe under most possible scenarios, including power loss, runaway motor, and control system. However, there is a remote possibility that these systems, even with their built-in redundancies, could fail. There is a remote possibility of injury (if all the safety valves (two mechanical and two electronic safety valves), and software safety checking program fail at the same time) by applying too high pressure in G-suit and PPB. The risks of PPB are well known, and documented elsewhere [2]. Minor risks include petechial hemorrhage, loss of central blood volume and fainting, breathing discomfort and facial discomfort. The pressure from the G-suit may cause limb tingling. More serious but rare complications can include fainting, pulmonary barotrauma resulting from inadequate thoracic counter-pressurization, with introduction of air into the arterial circulation, and cardiac arrhythmia. Minimization of these risks is ensured by proper medical screening, a redundant fail-safe pressure delivery system, and the wearing of properly fitted Anti-G-suits and thoracic counterpressure garments.

I understand that a medical interview, examination, and tests will be required before my participation begins I hereby consent to the medical assessment outlined in the protocol and agree to provide responses to questions that are to the best of my knowledge, truthful and complete. Before I begin any experimental sessions, I must inform the investigator of any changes to my medical status since my initial assessment including bit not limited to viral illnesses, new prescription or "over-the-counter" medications, and new risk of pregnancy. I have been advised that the medical information I reveal will be treated as confidential ('Protected B' IAW CF Security Requirements) and not revealed to anyone other than the investigator without my consent except as data unidentified as to source. I understand and agree that should I require medical treatment as a result of this study, this will be provided or coordinated by the physician assigned to this study. I understand that a medical officer will not attend the experimental sessions, but will be on-call and in the immediate vicinity in case of emergencies, and hereby consent to whatever emergency medical intervention deemed necessary by the attending medical personnel or their consultants.

I am aware that I will participate in an experimental session of 3 hours length (including subject preparation time) involving approximately 26 individual runs consisting of G-suit pressurization, PPB schedules, and tilt-table rotation while being monitored by a number of cardiovascular and cerebrovascular instruments as outlined in the protocol.

I have been assured that any personal information concerning me that is revealed in connection with this study will be kept in strict confidence except as data unidentified as to source.

I agree to: (i) not consume any alcoholic beverages within 24 hrs of each series of experiments, (ii) have a restful night evening the night before a test; (iii) perform no heavy exercise on the day of a test; and (iv) abstain from smoking, coffee, cola beverages and chocolate in the 3 hr period before a test, and a meal in the hour preceding it. I understand that a blood donation less than 30 days in advance of commencement of experimentation are contraindications to my participation in the study

For female subjects: I have been informed that that this experimental protocol could be potentially harmful to a fetus. Therefore I consent to the administration of a pregnancy (serum/blood) test as part of the medical procedure 48 hours prior to the commencement of this experiment. I understand that this result and all discussion pertaining to this matter will be treated as confidential between the physician and subject. If I have concern regarding a possible pregnancy I will consult a DCIEM physician before undertaking or resuming any phase of the experiment. Furthermore, I will take appropriate precautions to prevent pregnancy for the duration of the experiment

This consent is voluntary and has been given under circumstances in which I can use free power of choice. I have been informed that I may, at any time, revoke my consent and withdraw from the experiment. In the event of a loss of consciousness episode as a result of PPB exposure, I shall refrain from driving or operative heavy machinery for a period of 4 hrs following the experimental session. If transportation is required because of this recommendation, the principal investigator(s) shall make such arrangements, the cost of which, where necessary, will be borne by DCIEM. Also, the investigators, their designate, or the physician(s) responsible for the experiment may terminate the experiment at any time, regardless of my wishes.

The medical advisor or another DCIEM flight surgeon will be available for any medical questions and concerns prior to the experiment and should be the first person contacted if any medical problems or concerns arise following the experiment. The Principle Investigator will be available prior to the experiment for any questions dealing with the experimental protocol

Volunteer Name ______________________ Signature, ________________ Date __________

Witness Name ______________________ Signature ________________ Date __________

This subject is certified fit to participate in this experiment as outlined in the protocol with the limitations appended below.

Physician/Med Off. ______________________ Signature ________________ Date __________

Section Head/Commanding Officers
Name ______________________ Signature ________________ Date __________

Principal Investigator. ______________________ Signature ________________ Date __________

DCIEM Runs for Experimental Protocol #L239

Evaluation of a Subject's Physiological Responses to Nonlinear G-suit, Positive Pressure Breathing Schedules, and Straining during Low Intensity Push-Pull Transitions
Development of an Anti-G Protection Expert System (AGPES)

Tilt-Table Experiment
Development of an Anti-G Protection Expert System (AGPES)
(Protocol #L239)

January 2000

Subject Name :................................

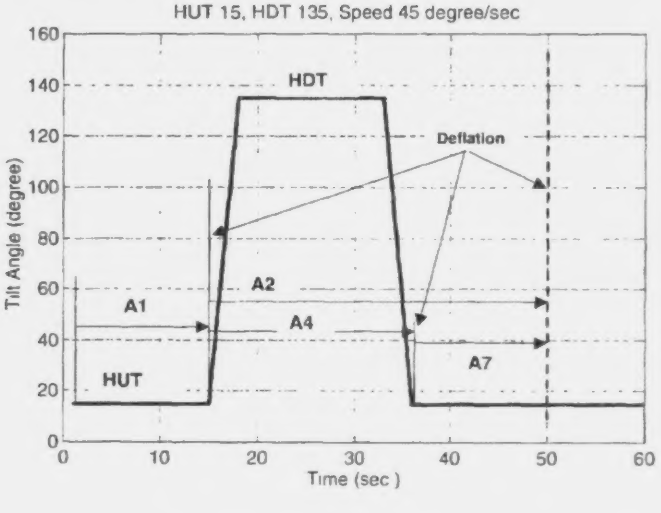
Date :.........................

DCIEM-XI Runs for Experimental Protocol# L239

Subject Name: **Date:**

1. Pre-experiment Data			
▪ **Experience** (hrs.)	Flight	2. Centrifuge	3. Tilt-table
▪ **Physical exercise / Fitness**			
4. What type?		5. How often?	
▪ **In the past 12 hours:**			
Tobacco use	6. Y/N	Caffeine consumption	7. Y/N
Alcohol consumption	Y/N	Exercise	Y/N
Back ache	Y/N	Nausea	Y/N
Congestion problems	Y/N	Transfusion, medication	Y/N
Diving, flying, chamber runs	Y/N	Last night's sleep	Normal/more/less
▪ **In the past 4 hours:**		8. Chocolate consumption	9. Y/N
▪ **In the past 2 hours:**		10. Food consumption	11. Y/N
Subject Data			
Gender	12. M/F	Total height (in)	
Age		Height to eye midpoint From heart level (in)	
Weight (lb.)		Vision	13.
Helmet Size	14. S/M/L	G-suit size	15. S/M/L
[redacted]			

Inflation Schedules:



DCIEM-XI Runs for Experimental Protocol# L239

Protocol #L239 Experiments				
Test #	Test Filename	G-suit Profile	PPB Profile	Remarks
1	Relax1	N/A	N/A	-
2	Control1-1	N/A	N/A	-
3	16. Control2	N/A	N/A	-
4	Relax2	N/A	N/A	-
5	TGS-CO	TGS-CO	N/A	4 psi
6	TGS-A7	TGS-A7	N/A	4 psi
7	TGS-A1	TGS-A1	N/A	4 psi
8	TPP-CO-1	N/A	TPP-CO-1	20 mmHg
9	TPP-CO-2	N/A	TPP-CO-2	30 mmHg
10	TPP-A4-1	N/A	TPP-A4-1	20 mmHg
11	TPP-A4-2	N/A	TPP-A4-2	30 mmHg
12	TPP-A4-3	N/A	TPP-A4-3	30 mmHg
13	TPP-A2	N/A	TPP-A2	30 mmHg
14	TPG-CO-1	TGS-CO	TPP-CO	30 mmHg, 4 psi
15	TPG-CO-2	TGS-CO	TPP-CO	50 mmHg, 5. psi
16	TPGA77-1	TGS-A7	TPP-A7	30 mmHg, 4 psi
17	TPGA77-2	TGS-A7	TPP-A7	50 mmHg, 5. psi
18	TPG-A47	TGS-A7	TPP-A4	30 mmHg, 4 psi
19	TPG-A27	TGS-A7	TPP-A2	30 mmHg, 4 psi
20	TPG-A411	TGS-A1	TPP-A4	20 mmHg, 4 psi
21	TPG-A412	TGS-A1	TPP-A4	30 mmHg, 4 psi
22	TST-CO	N/A	N/A	-
23	Relax3	N/A	N/A	-
24	TST-A1	N/A	N/A	-
25	Control1-2	N/A	N/A	-
26	Relax4	N/A	N/A	-

Physiological Measurements, Techniques and Instrumentation:

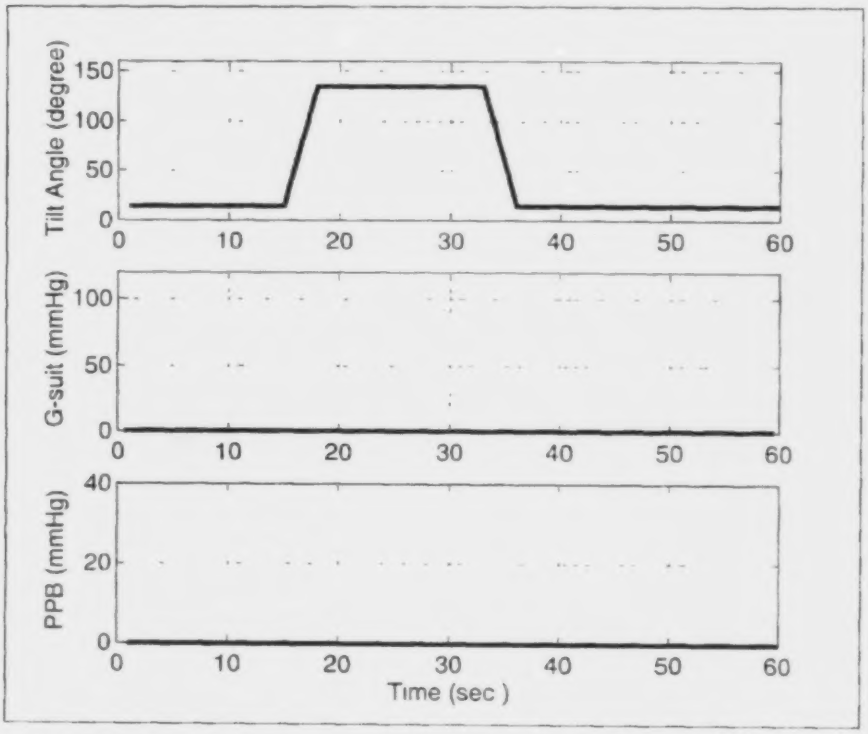
- Heart rate (electrocardiograph);
- Blood pressure (portapres) head-level & heart level;
- Ear opacity (Nonnin);
- Breathing tone (respitrace).
- Peripheral Light (Light bar)

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 1 Relax1

Test No. 2 Control1-1

Total Time: 1 min. Max G-suit Pressure. 0 psi (0 mmHg) Max PPB. 0 mmHg
HUT. 15° HDT. 135° Tilt Speed: 45°/sec.



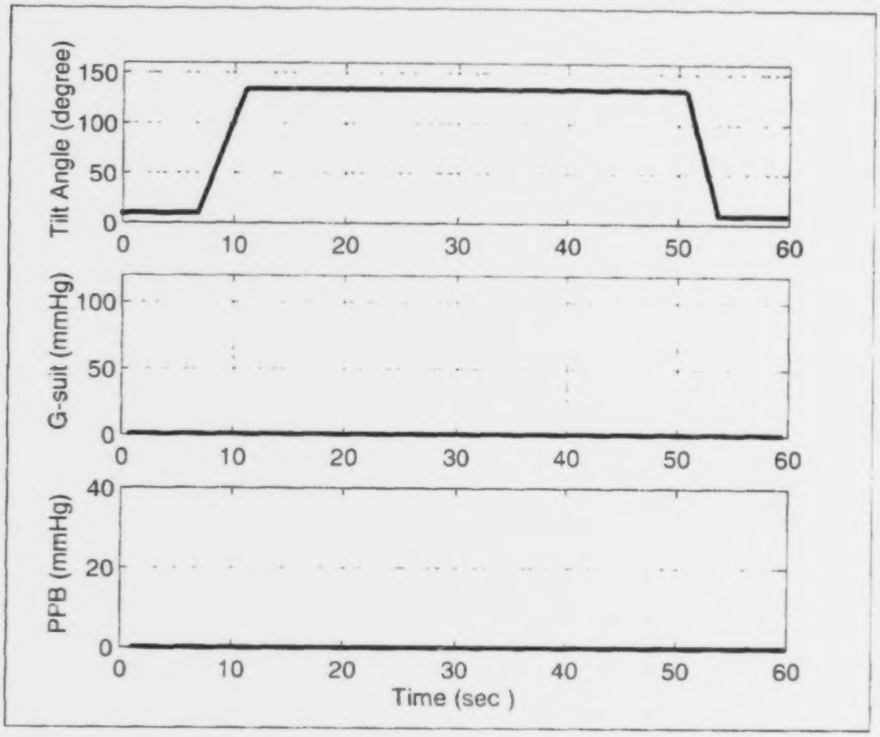
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify·

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 3 Control2

Total Time: 2 min. Max G-suit Pressure 0 psi (0 mmHg) Max PPB. 0 mmHg
HUT: 15° HDT. 135° Tilt Speed. 45°/sec.



Comments:

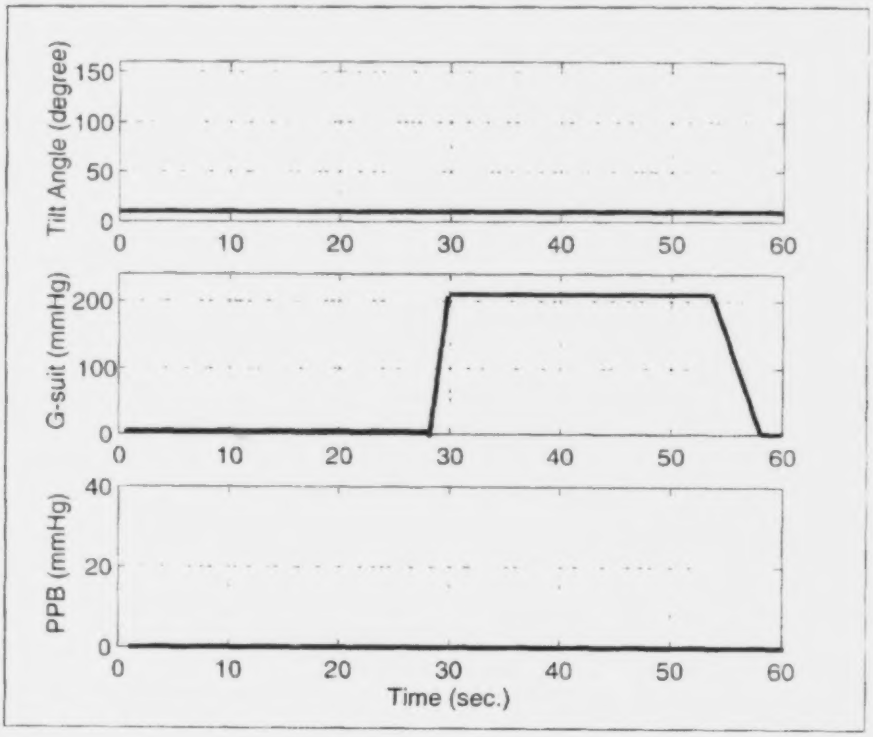
Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify:

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 4 Relax2

Test No. 5 TGS-CO

Total Time: 1 min.	Max G-suit Pressure: 4 psi (206.8 mmHg)	Max PPB: 0 mmHg
HUT: 15° HDT: 15°	Tilt Speed: 45°/sec.	



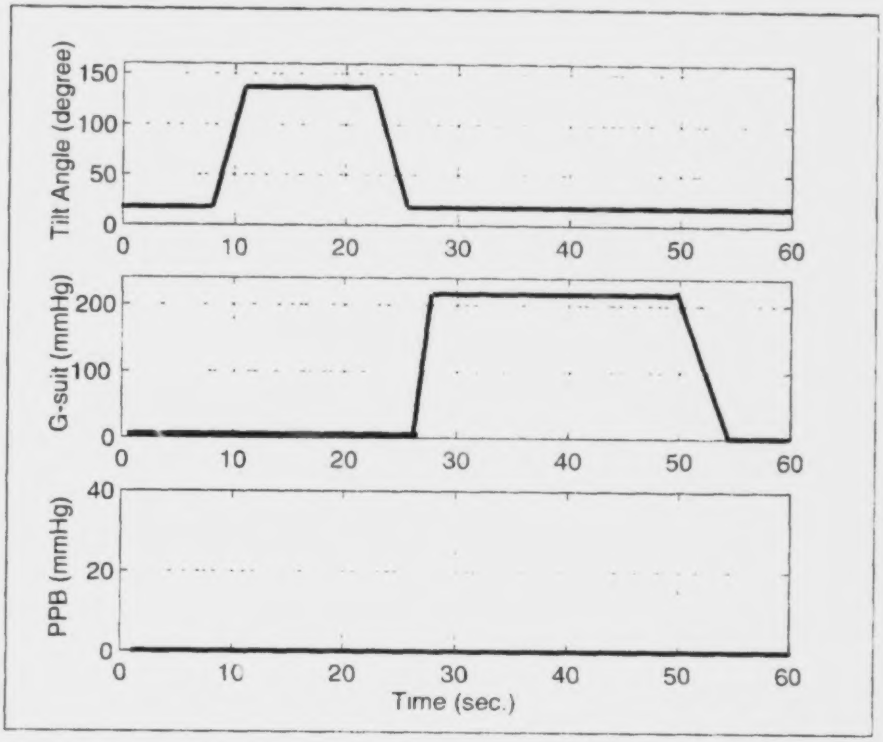
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 6 TGS-A7

Total Time: 1 min. Max G-suit Pressure: 4 psi (206.8 mmHg) Max PPB: 0 mmHg
HUT. 15° HDT: 135° Tilt Speed: 45°/sec



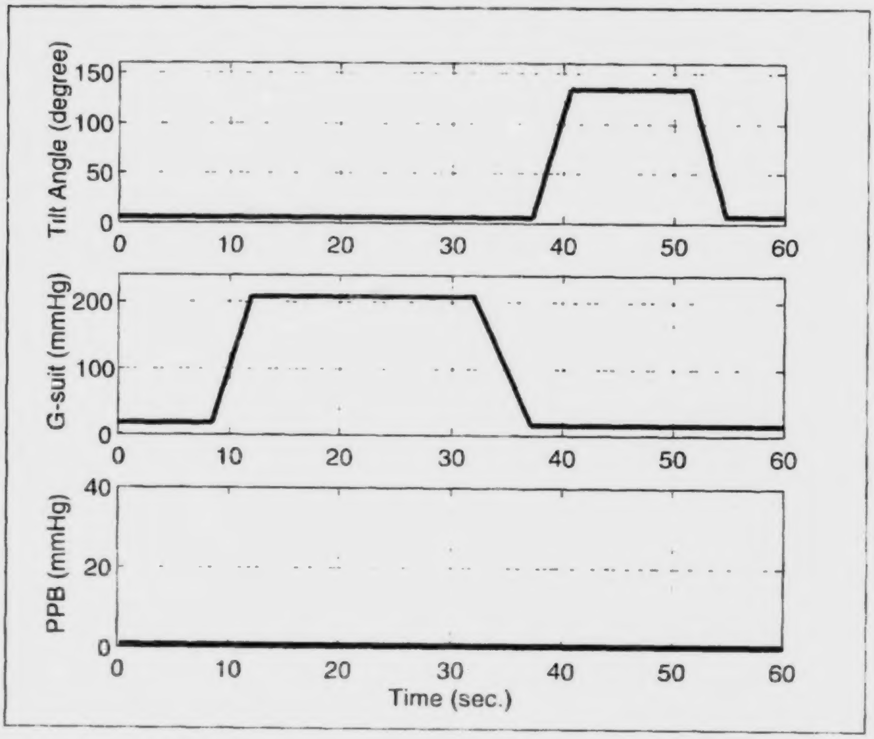
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify.

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 7 TGS-A1

Total Time: 1 min Max G-suit Pressure: 4 psi (206 8 mmHg) Max PPB. 0 mmHg
HUT· 15° HDT. 135° Tilt Speed. 45°/sec.



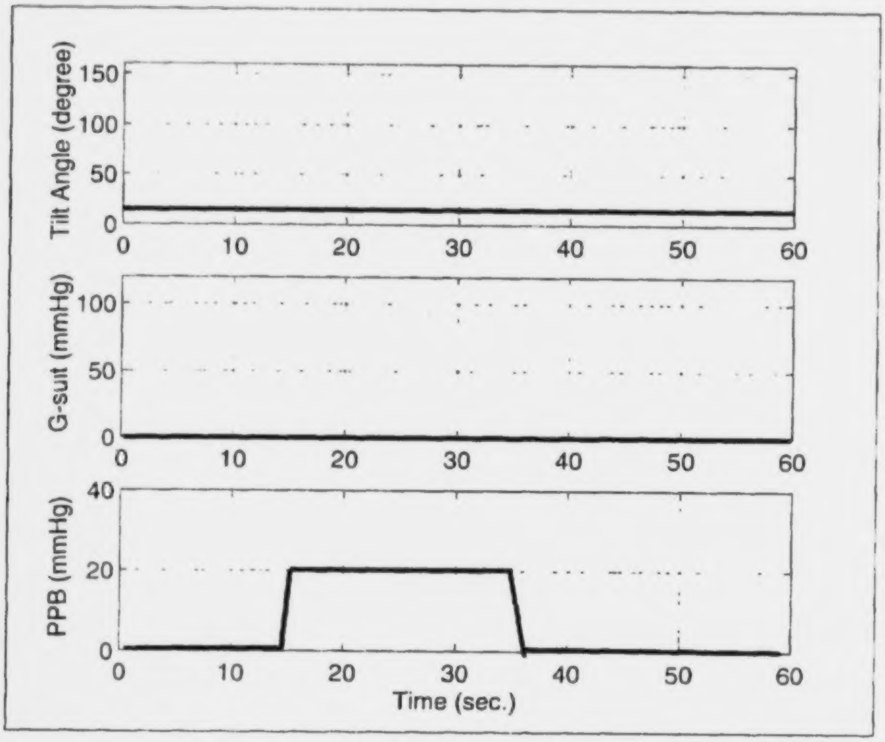
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 8 TPP-CO-1

Total Time: 1 min.	Max G-suit Pressure: 0 psi (0 mmHg)	Max PPB: 20 mmHg
HUT: 15°	HDT. 15°	Tilt Speed: 0°/sec.



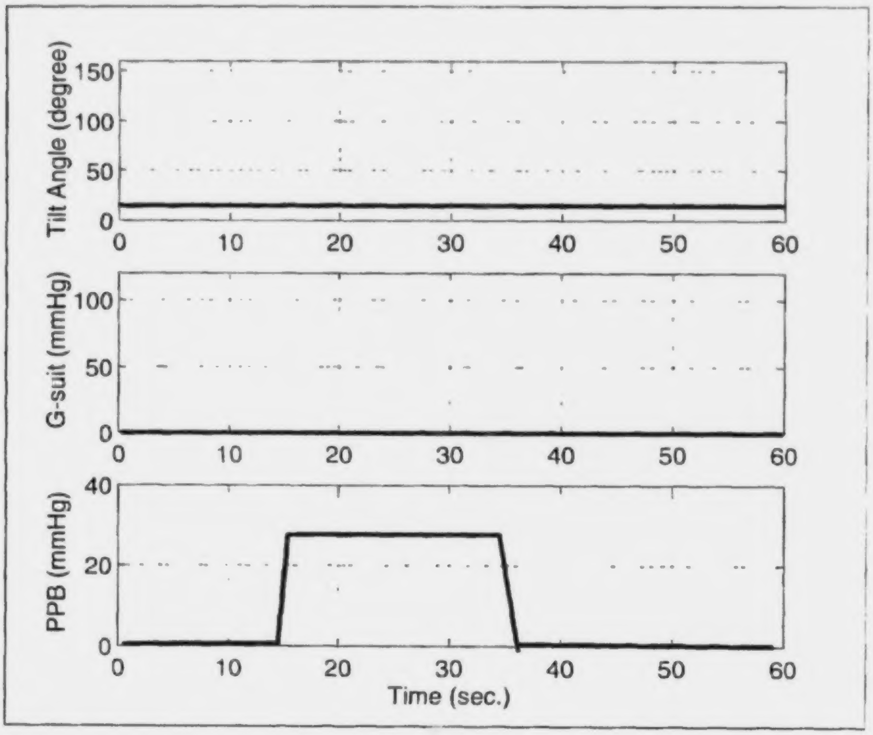
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 9 TPP-CO-2

Total Time: 1 min. Max G-suit Pressure. 0 psi (0 mmHg) Max PPB: 30 mmHg
HUT: 15° HDT. 15° Tilt Speed: 0°/sec.



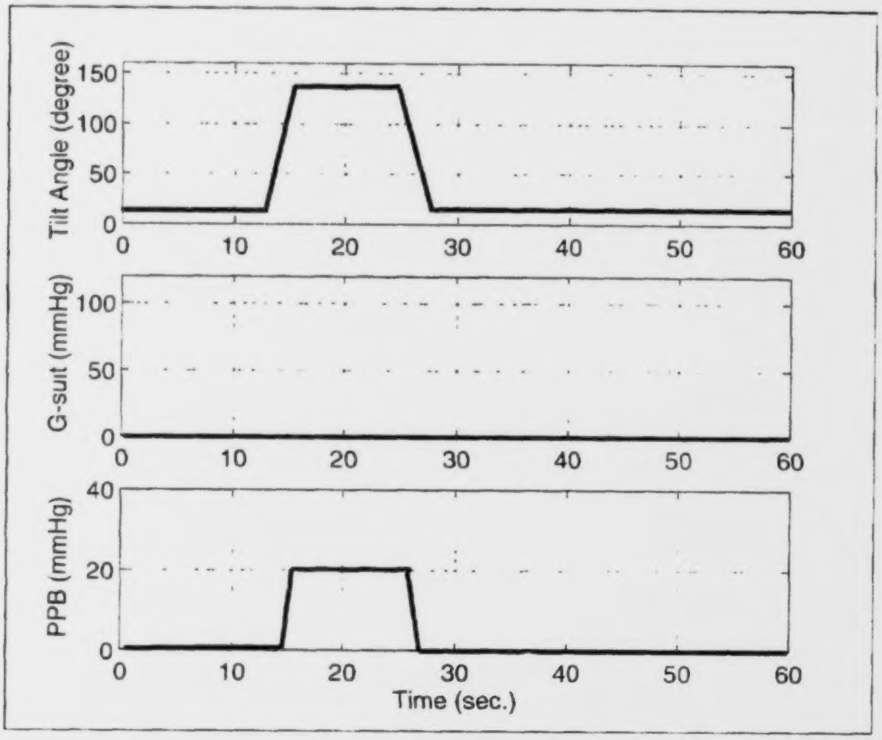
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify:

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 10 TPP-A4-1

Total Time: 1 min Max G-suit Pressure: 0 psi (0 mmHg) Max PPB. 20 mmHg
HUT 15° HDT 135° Tilt Speed: 45°/sec.



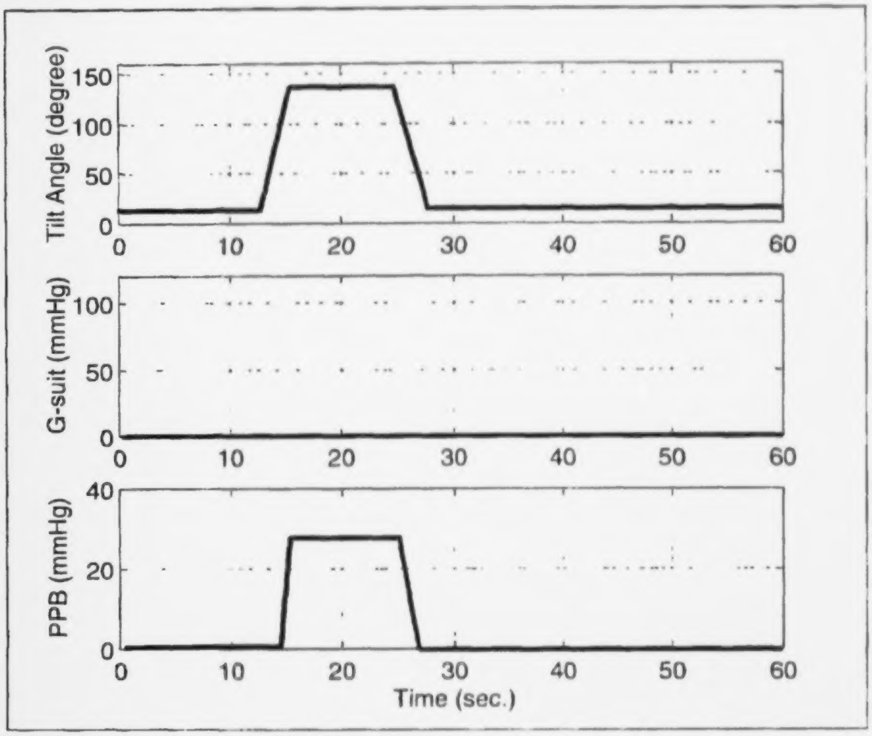
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify.

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 11 TPP-A4-2

Total Time: 1 min.	Max G-suit Pressure: 0 psi (0 mmHg)	Max PPB: 30 mmHg
HUT: 15°	HDT: 135°	Tilt Speed: 45°/sec.



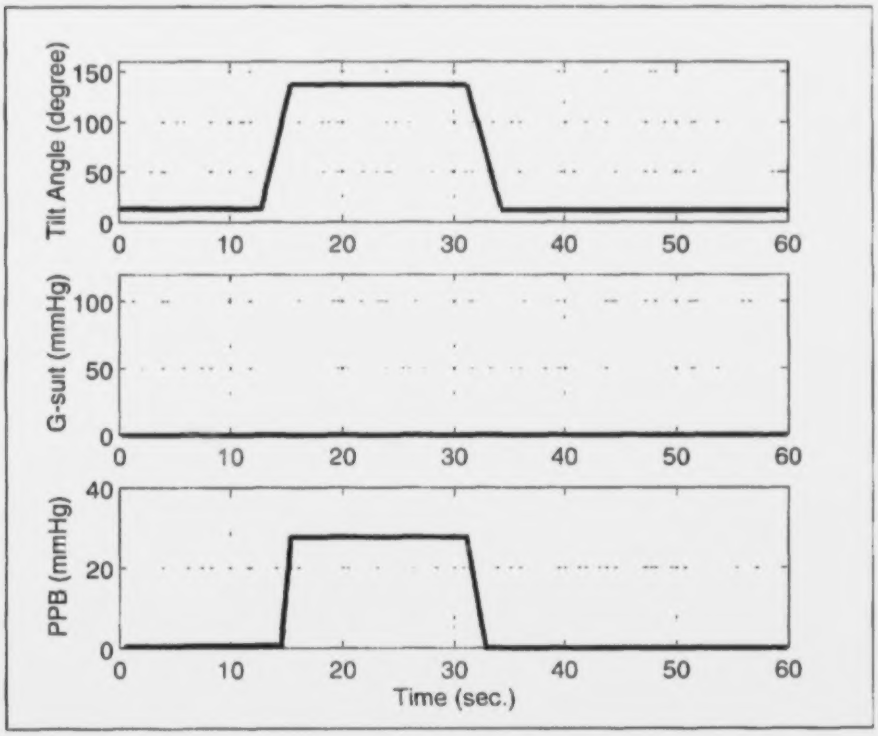
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify:

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 12 TPP-A4-3

Total Time: 1 min. Max G-suit Pressure. 0 psi (0 mmHg) Max PPB: 30 mmHg
HUT. 15° HDT: 135° Tilt Speed: 45°/sec



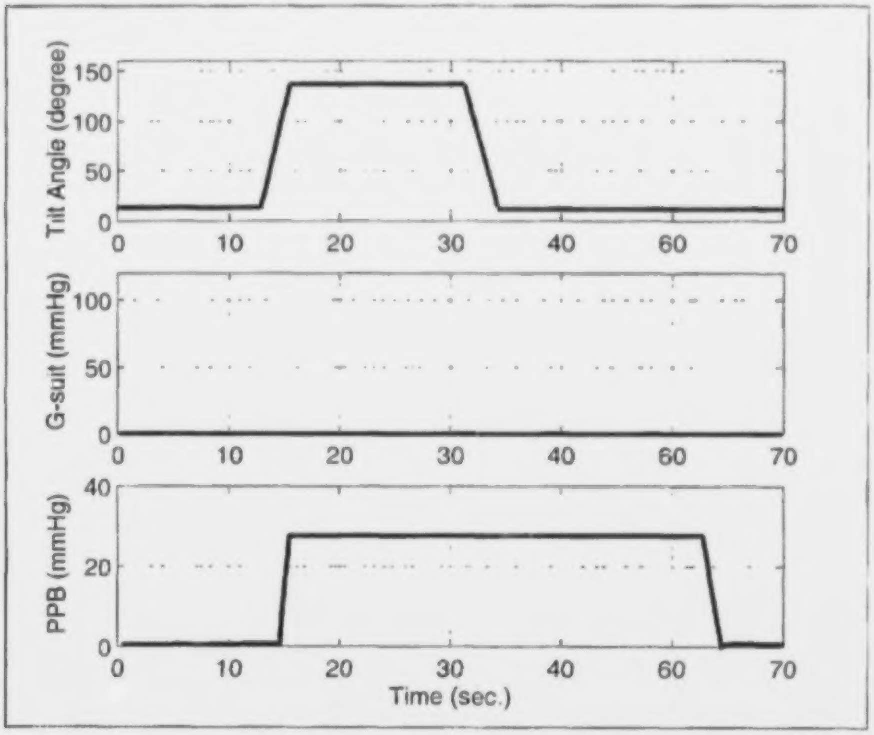
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 13 TPP-A2

Total Time: 1 min Max G-suit Pressure: 0 psi (0 mmHg) Max PPB: 30 mmHg
HUT: 15° HDT: 135° Tilt Speed: 45°/sec.



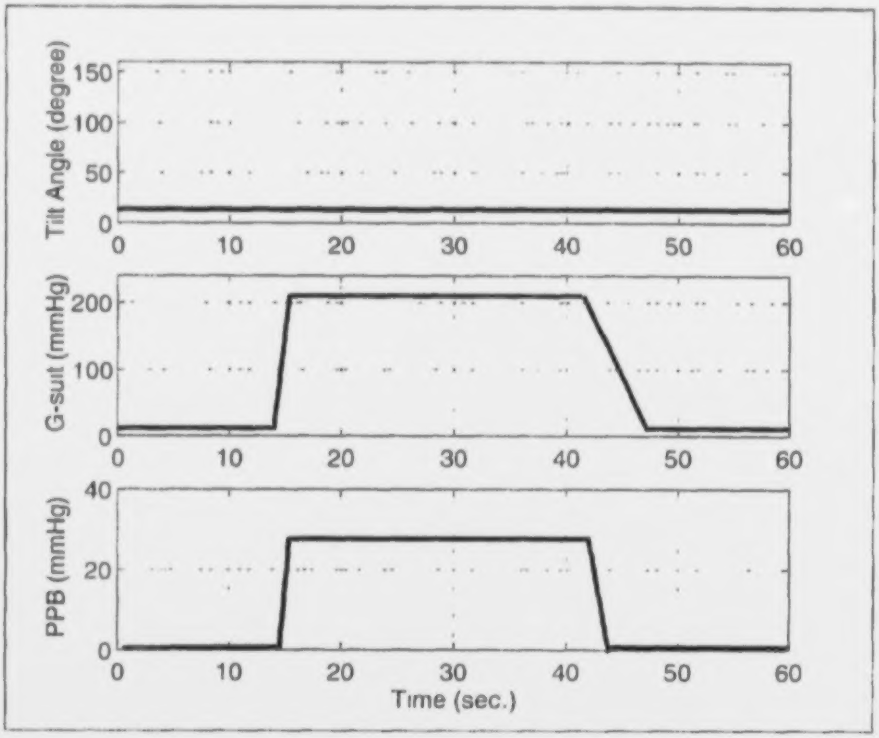
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify.

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 14 TPG-CO-1

Total Time 1 min.	Max G-suit Pressure. 4 psi (206.8 mmHg)	Max PPB 30 mmHg
HUT: 15° HDT: 15°	Tilt Speed: 0°/sec.	



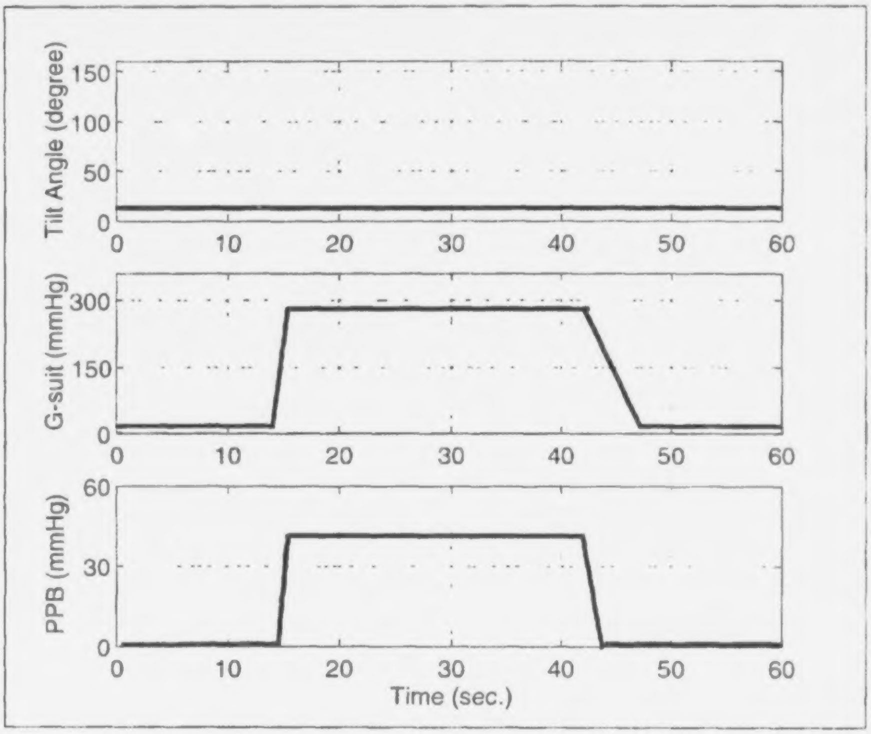
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify.

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 15 TPG-CO-2

Total Time 1 min. Max G-suit Pressure· 5.25 psi (271 4 mmHg) Max PPB: 50 mmHg
HUT: 15° HDT· 15° Tilt Speed· 0°/sec



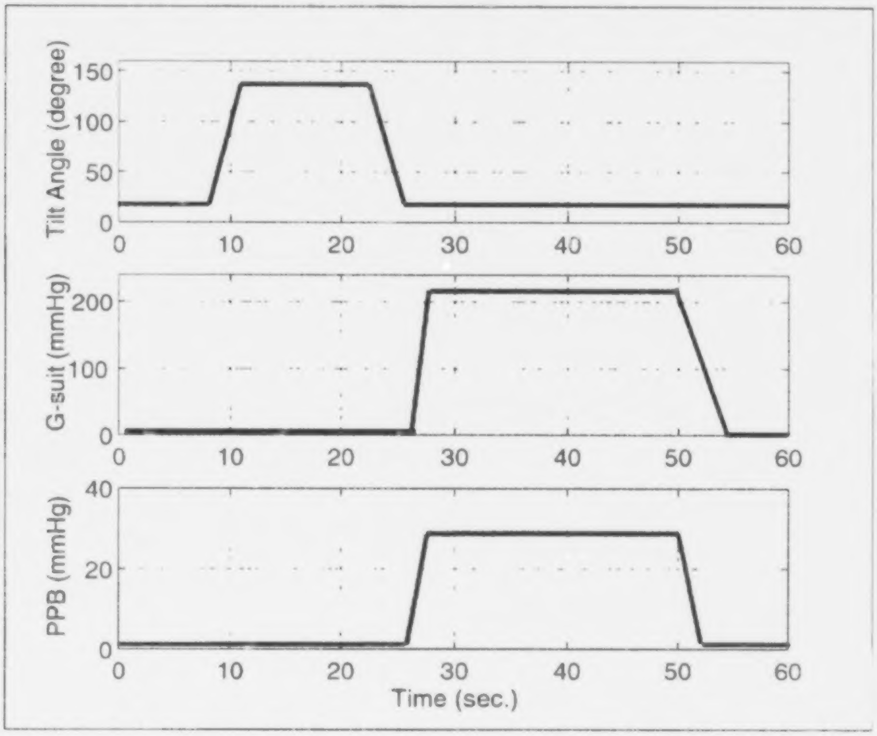
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify·

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 16 TPG-A77-1

Total Time. 1 min. Max G-suit Pressure 4 psi (206.8 mmHg) Max PPB: 30 mmHg
HUT: 15° HDT: 135° Tilt Speed. 45°/sec.



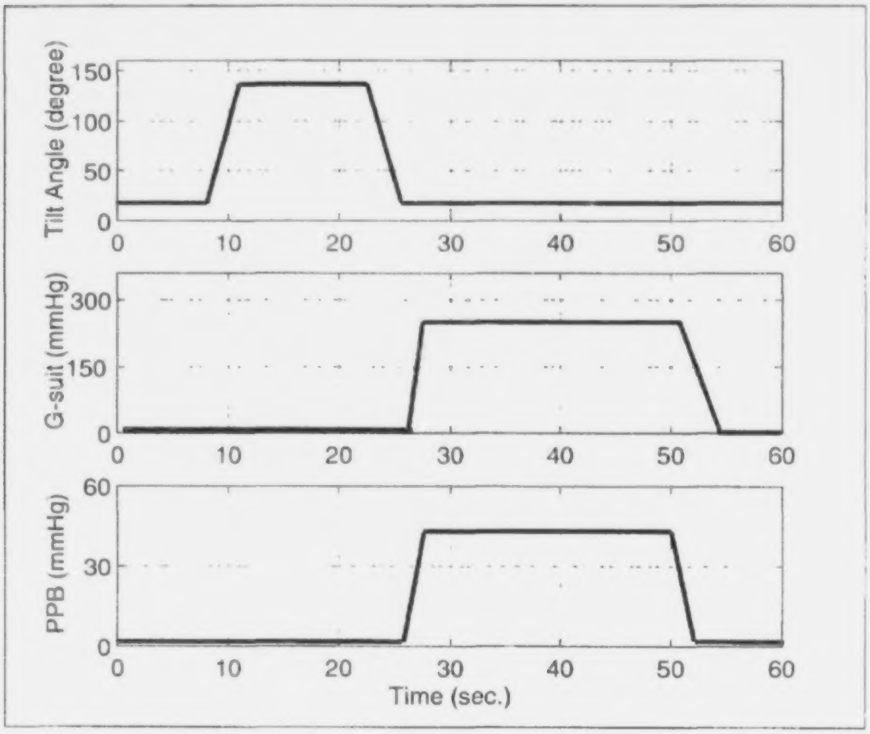
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 17 TPG-A77-2

Total Time: 1 min. Max G-suit Pressure. 5.25 psi (271 4 mmHg) Max PPB: 50 mmHg
HUT 15° HDT: 135° Tilt Speed. 45°/sec.



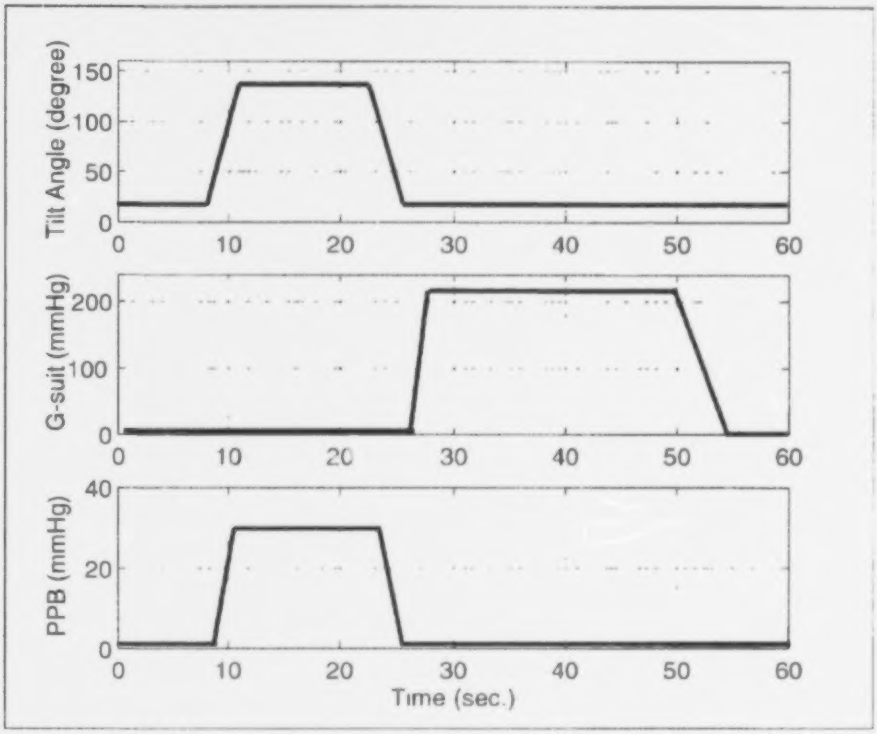
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify:

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 18 TPG-A47

Total Time: 1 min. Max G-suit Pressure: 4 psi (206 8 mmHg) Max PPB: 30 mmHg
HUT. 15° HDT: 135° Tilt Speed· 45°/sec.



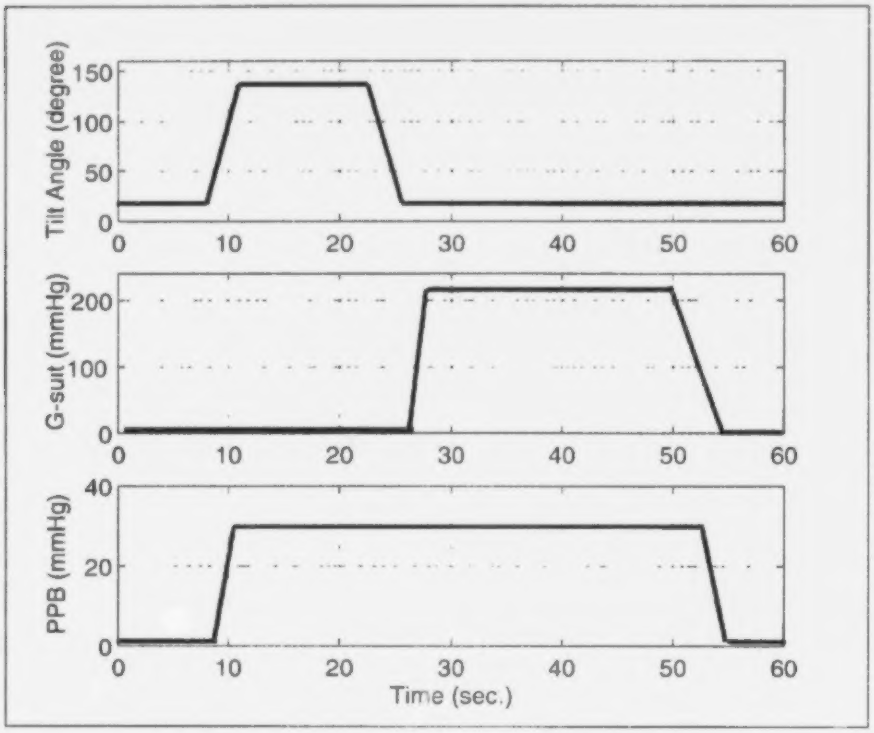
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci.	Other	Specify

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 19 TPG-A27

Total Time: 1 min. Max G-suit Pressure: 4 psi (206.8 mmHg) Max PPB: 30 mmHg
HUT: 15° HDT: 135° Tilt Speed 45°/sec.



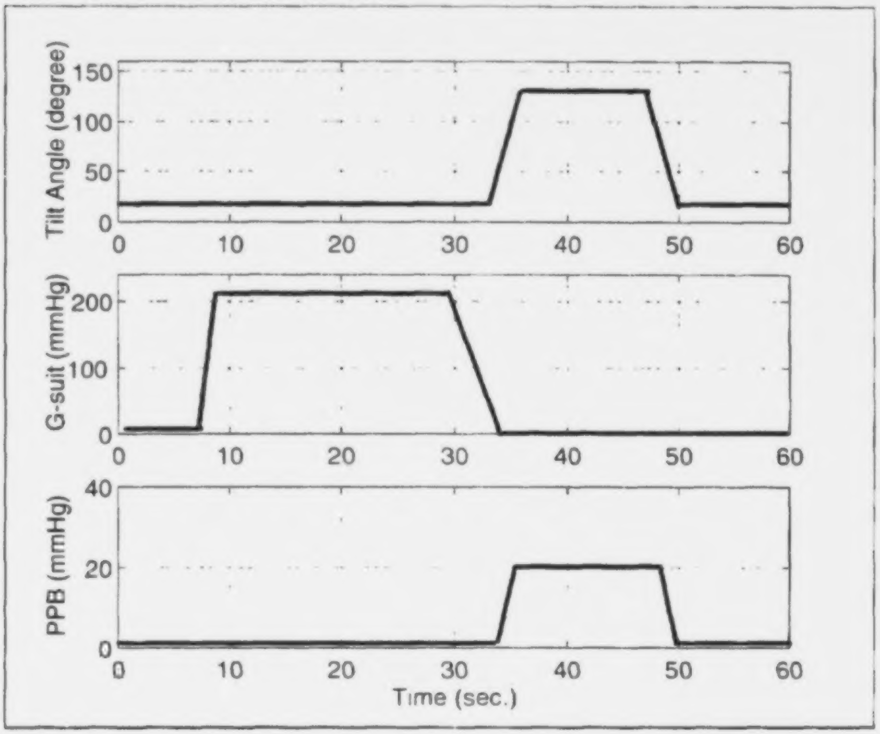
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify:

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 20 TPG-A41-1

Total Time. 1 min Max G-suit Pressure: 4 psi (206.8 mmHg) Max PPB: 20 mmHg
HUT: 15° HDT: 135° Tilt Speed: 45°/sec.



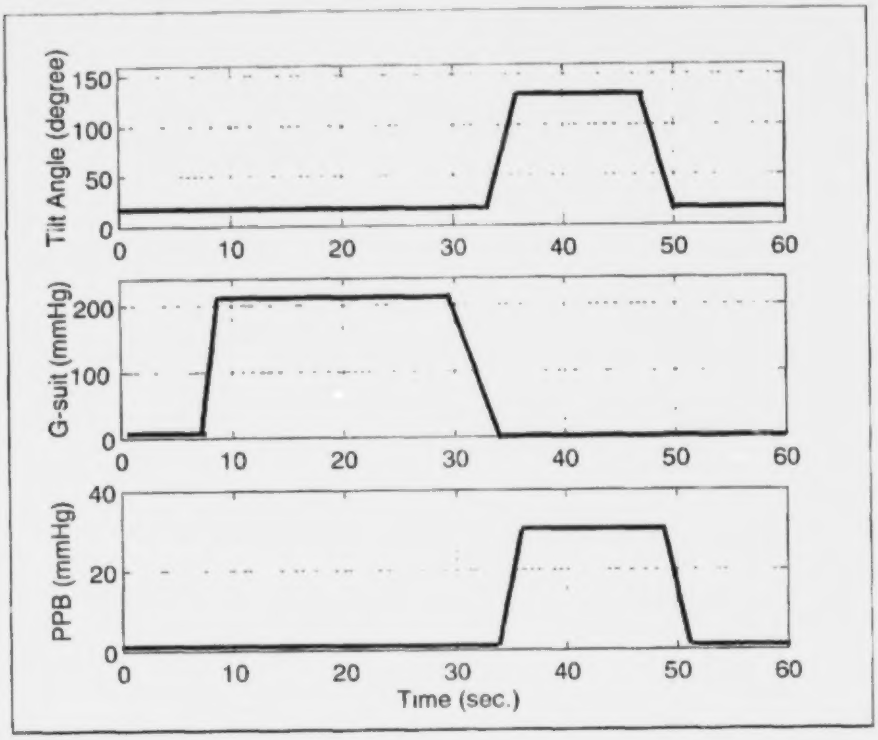
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify:

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 21 TPG-A41-2

Total Time: 1 min.	Max G-suit Pressure: 4 psi (206.8 mmHg)	Max PPB: 30 mmHg
HUT: 15°	HDT. 135°	Tilt Speed: 45°/sec.



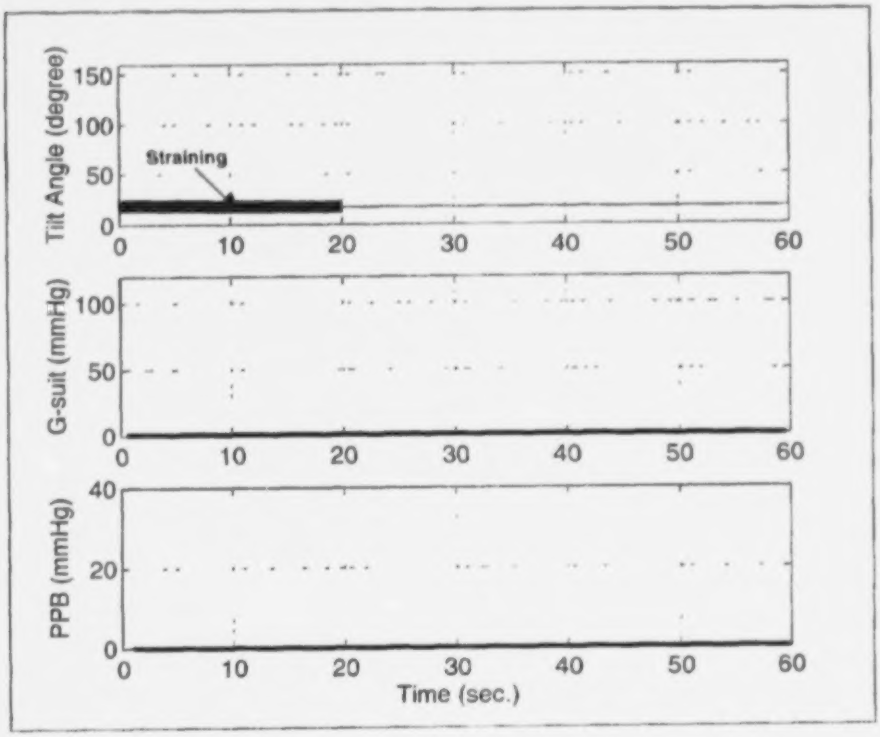
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 22 TST-CO

Total Time: 1 min. Max G-suit Pressure. 0 psi (0 mmHg) Max PPB· 0 mmHg
HUT: 15° HDT: 135° Tilt Speed: 45°/sec.



Comments:

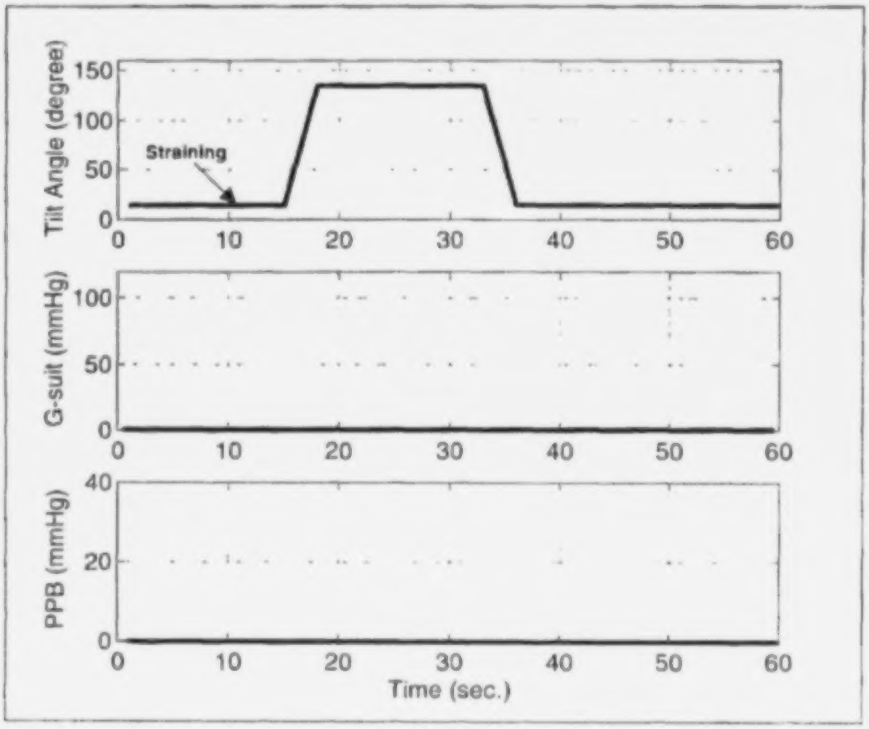
Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 23 Relax3

Test No. 24 TST-A1

Total Time: 1 min.	Max G-suit Pressure. 0 psi (0 mmHg)	Max PPB. 0 mmHg
HUT: 15°	HDT: 135°	Tilt Speed: 45°/sec.



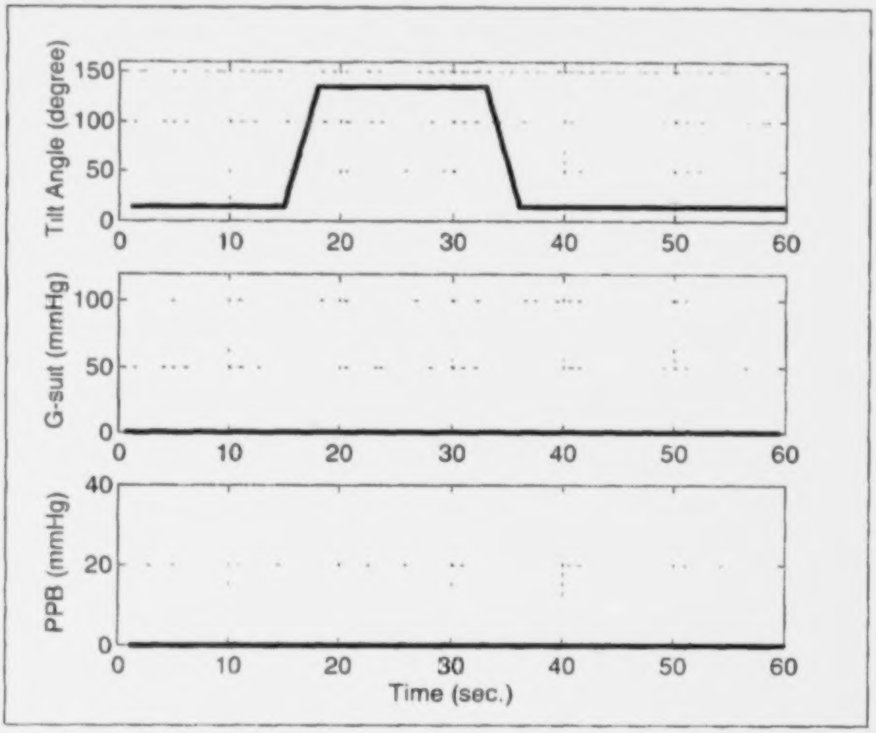
Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify

DCIEM-XI Runs for Experimental Protocol# L239

Test No. 25 Control1-2

Total Time: 1 min.	Max G-suit Pressure: 0 psi (0 mmHg)	Max PPB: 0 mmHg
HUT· 15°	HDT· 135°	Tilt Speed: 45°/sec.



Comments:

Vision (difference between Ini. & Final)	Yes	Minor	No	
Facial Congestion	Low	Med	High	
Comapre with previous one	Better	Same	Worst	
Gsuit Causes Pain	Yes		No	
PPB Causes Pain	Yes		No	
Straining	Yes		No	
Breathing	Normal		Abno	
Breathing	Deep		Shal	
Emotion	Nerv	Exci	Other	Specify:

Test No. 26 Relax4 END OF EXPERIMENT

DOCUMENT CONTROL DATA SHEET

1a PERFORMING AGENCY

Engineering Services Inc 1 Bedford Rd , Unit 5 Toronto, ON Canada M5R 2J7

2 SECURITY CLASSIFICATION

UNCLASSIFIED
Unlimited distribution -

1b PUBLISHING AGENCY

DCIEM

3 TITLE

(U) Test and Evaluation of CF-18 G-Valve Prototype

4 AUTHORS

Ziren Lu Matt Morel Alex Kapps Aaron Galluzzi

5 DATE OF PUBLICATION

October 29 , 2001

6. NO OF PAGES

245

7 DESCRIPTIVE NOTES

8 SPONSORING/MONITORING/CONTRACTING/TASKING AGENCY

Sponsoring Agency·
Monitoring Agency
Contracting Agency
Tasking Agency DCIEM

9 ORIGINATORS DOCUMENT NO

Contract Report CR 2002-006

10 CONTRACT GRANT AND/OR PROJECT NO

W7711-007631

11 OTHER DOCUMENT NOS

12 DOCUMENT RELEASABILITY

Unlimited distribution

13 DOCUMENT ANNOUNCEMENT

Unlimited announcement

14 ABSTRACT

(U) This final report summarized the preliminary experimental testing program and statistical validation of novel countermeasures aimed at alleviating Push-Pull effects. The new countermeasures for active attenuation of Push-Pull effect involve applying mid-level Positive Pressure Breathing (PPB) during the Push phase in accordance with the specially formulated non-linear pressure schedules. To formally investigate the proposed countermeasures, a custom-designed statistical modelling framework ESIstat has been integrated using R language.

It has been statistically demonstrated in Tilt-table experiments that the application of medium-intensity (30mmHg) PPB during -Gz followed with the application of G-suit pressure and regular-intensity PPB during the transition to +Gz attenuates the Push-Pull effect and may prevent the subsequent decrease in a pilot's G-tolerance due to -Gz exposure. If the new advanced tests planned on the upcoming Push-Pull centrifuge at DCIEM substantiate the proposed protection strategy, it might potentially constitute an efficient and practical Anti-G countermeasure. The new protection can be readily incorporated into the existing Anti-G ensemble by simply retrofitting the PPB mask/jerkin regulator.

15 KEYWORDS, DESCRIPTORS or IDENTIFIERS

(U) CF-18; G-Valves; push-pull effect; positive pressure breathing; PPB; +Gz; -Gz